

Panduan Pengguna

Amazon EKS



Amazon EKS: Panduan Pengguna

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Merek dagang dan tampilan dagang Amazon tidak boleh digunakan sehubungan dengan produk atau layanan apa pun yang bukan milik Amazon, dengan cara apa pun yang dapat menyebabkan kebingungan di antara pelanggan, atau dengan cara apa pun yang menghina atau mendiskreditkan Amazon. Semua merek dagang lain yang tidak dimiliki oleh Amazon merupakan hak milik masing-masing pemiliknya, yang mungkin atau tidak terafiliasi, terkait dengan, atau disponsori oleh Amazon.

Table of Contents

.....	xxii
.....	xxiii
Apa itu Amazon EKS?	1
Amazon EKS: Manajemen Kubernetes Sederhana	1
Fitur Amazon EKS	2
Layanan terkait	3
Harga Amazon EKS	4
Kasus penggunaan umum	5
Arsitektur	6
Bidang kontrol	6
Hitung	7
Konsep Kubernetes	9
Mengapa Kubernetes?	9
Klaster	14
Beban kerja	19
Langkah selanjutnya	25
Opsi deployment	26
Memahami opsi penerapan Amazon EKS	26
Amazon EKS di cloud	26
Amazon EKS di pusat data atau lingkungan edge	27
Amazon EKS Anywhere untuk lingkungan yang memiliki jurang udara	28
Perkakas Amazon EKS	29
Penyiapan	30
Langkah selanjutnya	30
Mengatur AWS CLI	30
Untuk membuat kunci akses	31
Untuk mengkonfigurasi AWS CLI	31
Untuk mendapatkan token keamanan	32
Untuk memverifikasi identitas pengguna	32
Langkah selanjutnya	33
Siapkan kubectl dan eksctl	33
Instal atau perbarui kubectl	33
Langkah 1: Periksa apakah kubectl sudah diinstal	33
Langkah 2: Instal atau perbarui kubectl	34

Instal eksctl	47
Langkah selanjutnya	48
Mulai cepat	49
Dalam tutorial ini	49
Prasyarat	49
Konfigurasikan cluster	50
Buat IngressClass	51
Menyebarkan aplikasi sampel game 2048	51
Pertahankan Data menggunakan Mode Otomatis Amazon EKS	55
Hapus cluster dan node Anda	57
Pelajari Amazon EKS	58
Gambaran Umum	58
Lokakarya Amazon EKS	59
Tutorial penyiapan kluster langsung Amazon EKS	60
Sampel Amazon EKS	61
AWS Tutorial	62
Lokakarya Pengembang	62
Lokakarya Terraform	62
AWS Pelatihan Amazon EKS	62
Memulai	64
Buat cluster (Mode Otomatis EKS)	64
Buat cluster (eksctl)	65
Prasyarat	66
Langkah 1: Buat klaster dan simpul Amazon EKS Anda	66
Langkah 2: Lihat sumber daya Kubernetes	67
Langkah 3: Hapus klaster dan simpul Anda	69
Langkah selanjutnya	69
Buat cluster (Konsol dan CLI)	70
Prasyarat	70
Langkah 1: Buat klaster dan Amazon EKS Anda	71
Langkah 2: Konfigurasikan komputer Anda agar bisa berkomunikasi dengan klaster Anda	74
Langkah 3: Buat node	75
Langkah 4: Lihat sumber daya	77
Langkah 5: Hapus sumber daya	77
Langkah selanjutnya	78
Mode Otomatis EKS	80

Fitur	80
Komponen Otomatis	81
Konfigurasi	83
Buat cluster	83
CLI eksctl	84
AWS CLI	86
Konsol manajemen	94
Aktifkan cluster yang ada	96
Referensi migrasi	97
Migrasi volume EBS	98
Migrasi penyeimbang beban	99
Aktifkan di cluster	99
Migrasi dari Karpenter	103
Migrasi dari MNGs	107
Migrasi dari Fargate	107
Jalankan beban kerja	113
Menyebarkan beban kerja yang mengembang	114
Menyebarkan penyeimbang beban	117
Menyebarkan beban kerja stateful	122
Konfigurasi	127
Buat kelas simpul	130
Buat kumpulan simpul	138
Buat kelas ingress	144
Buat layanan	151
Buat StorageClass	157
Nonaktifkan Mode Otomatis EKS	166
Perbarui versi Kubernetes	168
Tinjau kumpulan node bawaan	169
Kontrol penyebaran	171
Jalankan add-on penting	173
Gunakan kebijakan jaringan	175
Tag subnet	176
Menyebarkan beban kerja yang dipercepat	178
Hasilkan laporan CIS	185
Enkripsi volume root	188
Perbarui kontrol AMI	192

Kontrol Penerapan ODCR	195
Cara kerjanya	196
Instans terkelola	197
Identitas dan akses	201
Jaringan	207
Pemecahan Masalah	210
Agen pemantauan simpul	211
Dapatkan output konsol dari instance EC2 terkelola dengan menggunakan AWS EC2 CLI ..	211
Dapatkan log node dengan menggunakan wadah debug dan CLI kubectl ..	212
Melihat sumber daya yang terkait dengan Mode Otomatis EKS di AWS Konsol	213
Lihat Kesalahan IAM di akun Anda AWS	213
Memecahkan masalah Pod yang gagal menjadwalkan ke node Mode Otomatis	214
Memecahkan masalah node yang tidak bergabung dengan cluster	214
Berbagi Volume di Seluruh Pod	217
Memecahkan masalah pengontrol yang disertakan dalam Mode Otomatis	217
Catatan rilis	218
Juni 30, 2025	218
Juni 20, 2025	218
Juni 13, 2025	218
April 30, 2025	218
April 18, 2025	219
April 11, 2025	219
April 4, 2025	219
Maret 31, 2025	219
Maret 21, 2025	219
Maret 14, 2025	219
Klaster	220
Buat kluster otomatis	221
Prasyarat	49
Buat cluster - AWS konsol	223
Buat cluster - AWS CLI	227
Langkah selanjutnya	94
Membuat klaster	233
Prasyarat	49
Langkah 1: Buat peran IAM cluster	235
Langkah 2: Buat cluster	236

Langkah 3: Perbarui kubeconfig	245
Langkah 4: Pengaturan cluster	246
Langkah selanjutnya	94
Wawasan cluster	248
Bersiaplah untuk upgrade versi Kubernetes dengan wawasan klaster	249
Memecahkan masalah kesalahan konfigurasi dengan wawasan klaster	250
Lihat wawasan klaster	250
Perbarui versi Kubernetes	255
Pertimbangan untuk Mode Otomatis Amazon EKS	257
Ringkasan	257
Langkah 1: Bersiaplah untuk upgrade	258
Langkah 2: Tinjau pertimbangan peningkatan	258
Langkah 3: Perbarui bidang kontrol cluster	260
Langkah 4: Perbarui komponen cluster	262
Turunkan versi Kubernetes untuk klaster Amazon EKS	264
Hapus klaster	264
Pertimbangan	99
Hapus cluster (eksctl)	265
Hapus cluster (AWS konsol)	266
Hapus cluster (AWS CLI)	267
Akses titik akhir klaster	268
IPv6format titik akhir cluster	269
IPv4format titik akhir cluster	269
Titik akhir pribadi cluster	270
Memodifikasi akses titik akhir klaster	271
Mengakses server API privat saja	273
Konfigurasi akses titik akhir	274
Aktifkan dukungan Windows	278
Pertimbangan	99
Prasyarat	49
Aktifkan dukungan Windows	280
Menyebarkan Pod Windows	282
Mendukung kepadatan Pod yang lebih tinggi pada node Windows	282
Nonaktifkan dukungan Windows	283
klaster privat	283
Versi Kubernetes	287

Versi yang tersedia pada dukungan standar	288
Versi yang tersedia pada dukungan diperpanjang	288
Kalender rilis Amazon EKS Kubernetes	288
Dapatkan informasi versi dengan AWS CLI	289
Versi Amazon EKS FAQs	291
Amazon EKS memperpanjang dukungan FAQs	293
Versi dukungan standar	295
Versi dukungan yang diperluas	301
Lihat periode dukungan	305
Lihat kebijakan peningkatan	305
Aktifkan dukungan yang diperluas	307
Nonaktifkan dukungan yang diperluas	309
Versi platform	310
Versi Kubernetes 1.33	311
Versi Kubernetes 1.32	312
Versi Kubernetes 1.31	314
Versi Kubernetes 1.30	317
Versi Kubernetes 1.29	320
Versi Kubernetes 1.28	325
Versi Kubernetes 1.27	331
Versi Kubernetes 1.26	337
Dapatkan versi platform saat ini	344
Ubah versi platform	344
Penskalaan otomatis	344
Mode Otomatis EKS	345
Solusi Tambahan	345
Pelajari tentang Zonal Shift	346
Memahami Arus Lalu Lintas Jaringan Timur-Barat Antar Pod	347
Persyaratan Pergeseran Zonal EKS	351
Pertanyaan yang Sering Diajukan	360
Sumber Daya Tambahan	363
Aktifkan Zonal Shift	363
Pertimbangan	364
Apa itu Pengontrol Pemulihan Aplikasi Amazon?	364
Apa itu pergeseran zona?	364
Apa itu zonal autoshift?	365

Apa yang dilakukan EKS selama autoshift?	365
Daftarkan cluster EKS dengan Amazon Application Recovery Controller (ARC) (AWS konsol)	365
Langkah Berikutnya	94
Kelola akses	367
Tugas Umum	367
Latar Belakang	176
Pertimbangan untuk Mode Otomatis EKS	264
Akses API Kubernetes	368
Kaitkan Identitas IAM dengan Izin Kubernetes	369
Atur Mode Otentikasi Cluster	370
Berikan izin	372
aws-auth ConfigMap	413
Tautkan penyedia OIDC	423
Putuskan tautan penyedia OIDC	428
Akses sumber daya klaster	428
Izin yang diperlukan	429
Akses klaster dengan kubectl	435
Buat kubeconfig file secara otomatis	436
Akses beban kerja ke AWS	437
Token akun layanan	438
Pengaya klaster	439
Memberikan izin AWS Identity and Access Management untuk beban kerja di klaster	
Amazon Elastic Kubernetes Service	440
Kredensial dengan IRSA	444
Identitas Pod	468
Kelola komputasi	502
Bandingkan opsi komputasi	502
Grup simpul terkelola	508
Konsep grup simpul terkelola	509
Tipe kapasitas grup simpul terkelola	512
Buat	515
Perbarui	526
Perbarui detail perilaku	531
Templat peluncuran	536
Hapus	552

Simpul yang dikelola sendiri	554
Amazon Linux	556
Bottlerocket	565
Windows	569
Ubuntu Linux	578
Metode pembaruan	581
AWS Fargate	594
AWS Pertimbangan Fargate	594
Tabel Perbandingan Fargate	597
Memulai	600
Tentukan profil	605
Hapus profil	610
Detail konfigurasi pod	611
Acara penambalan OS	614
Kumpulkan metrik	617
Pencatatan log	620
Jenis EC2 instans Amazon	631
Amazon EKS merekomendasikan Pod maksimum untuk setiap jenis EC2 instans Amazon .	633
Pertimbangan untuk Mode Otomatis EKS	264
Pra-dibangun dioptimalkan AMIs	635
AL2 Penghentian AMI	636
Amazon Linux	641
Bottlerocket	650
Ubuntu Linux	656
Windows	657
Kesehatan simpul	750
Agen pemantauan node	751
Perbaikan otomatis node	751
Masalah kesehatan simpul	752
Lihat kesehatan simpul	764
Dapatkan log simpul	767
Node hibrida	769
Fitur	770
Batas	771
Pertimbangan	771
Sumber daya tambahan	772

Prasyarat	772
Jalankan node hybrid	838
Konfigurasi	859
Cara kerjanya	898
Node hibrida nodeadm	939
Pemecahan Masalah	954
Penyimpanan data aplikasi	976
Amazon EBS	976
Pertimbangan	977
Prasyarat	977
Langkah 1: Buat peran IAM	978
Langkah 2: Dapatkan driver Amazon EBS CSI	985
Langkah 3: Menyebarkan aplikasi sampel	986
Amazon EFS	986
Pertimbangan	986
Prasyarat	987
Langkah 1: Buat peran IAM	988
Langkah 2: Dapatkan driver Amazon EFS CSI	994
Langkah 3: Buat sistem file Amazon EFS	994
Langkah 4: Menyebarkan aplikasi sampel	994
Amazon FSx untuk Lustre	995
Menyebarkan driver	995
Optimalkan (EFA)	1003
Optimalkan (Non-EFA)	1019
Amazon FSx untuk NetApp ONTAP	1026
Amazon FSx untuk OpenZFS	1027
Cache File Amazon	1028
Mountpoint untuk Amazon S3	1028
Pertimbangan	1028
Menyebarkan driver	1029
Hapus driver	1039
Pengontrol snapshot CSI	1041
Mengkonfigurasi jaringan	1042
Tambahkan Subnet VPC yang ada ke cluster Amazon EKS dari konsol manajemen	1042
Persyaratan VPC dan subnet	1042
Persyaratan dan pertimbangan VPC	1043

Persyaratan dan pertimbangan subnet	1045
Persyaratan dan pertimbangan subnet bersama	1051
Buat VPC	1052
Prasyarat	49
Subnet publik dan pribadi	1052
Hanya subnet publik	1055
Hanya subnet pribadi	1057
Persyaratan grup keamanan	1058
Grup keamanan klaster default	1058
Membatasi lalu lintas klaster	1060
Grup keamanan bersama	1061
Kelola add-on jaringan	1062
Pengaya bawaan	1062
Pengaya AWS jaringan opsional	1063
Amazon VPC CNI	1064
Plugin CNI alternatif	1177
AWS Pengontrol Load Balancer	1179
CoreDNS	1197
kube-proxy	1215
Beban kerja	1221
Contoh penyebaran (Linux)	1221
Prasyarat	49
Buat namespace	1222
Buat penerapan Kubernetes	1222
Membuat layanan	1224
Meninjau sumber daya yang dibuat	1225
Jalankan shell pada Pod	1228
Langkah Berikutnya	1229
Contoh penyebaran (Windows)	1229
Prasyarat	49
Buat namespace	1222
Buat penerapan Kubernetes	1222
Membuat layanan	1224
Meninjau sumber daya yang dibuat	1232
Jalankan shell pada Pod	1228
Langkah Berikutnya	1236

Vertical Pod Autoscaler	1236
Men-deploy Vertical Pod Autoscaler	1237
Uji pemasangan Vertical Pod Autoscaler	1238
Penskala Otomatis Pod Horizontal	1242
Jalankan aplikasi uji Penskala Otomatis Pod Horizontal	1243
Menyeimbangkan beban jaringan	1246
Prasyarat	49
Pertimbangan	99
Buat penyeimbang beban jaringan	1250
(Opsional) Deploy aplikasi sampel	1252
Penyeimbangan beban aplikasi	1256
Prasyarat	49
Gunakan kembali ALBs dengan Grup Ingress	1259
(Opsional) Deploy aplikasi sampel	1260
Batasi layanan eksternal IPs	1264
Salin gambar ke repositori	1266
Lihat pendaftar gambar Amazon	1269
Add-on Amazon EKS	1272
Pertimbangan	1273
Pertimbangan untuk Mode Otomatis Amazon EKS	1274
Dukungan	1275
AWS add-on	1277
Pengaya komunitas	1294
Pengaya Marketplace	1300
Buat add-on	1316
Perbarui add-on	1329
Verifikasi kompatibilitas	1337
Hapus add-on	1338
Peran IAM	1342
Bidang yang dapat Anda sesuaikan	1349
Verifikasi gambar kontainer	1352
Manajemen kluster	1354
Pemantauan biaya	1354
Lihat biaya berdasarkan Pod	1355
Instal Kubecost	1356
Akses Dasbor Kubecost	1357

Pelajari lebih lanjut tentang Kubecost	1358
Server metrik	1368
Pertimbangan	99
Terapkan sebagai add-on komunitas dengan Pengaya Amazon EKS	1369
Menyebarkan dengan manifes	1369
Menerapkan aplikasi dengan Helm	1370
Pemberian tag pada sumber daya Anda	1372
Dasar-dasar tag	1372
Pemberian tag pada sumber daya Anda	1373
Batasan tag	1374
Menandai sumber daya Anda untuk penagihan	1375
Bekerja dengan tanda menggunakan konsol	1376
Cara menggunakan tanda dengan menggunakan CLI, API, atau eksctl	1377
Kuota layanan	1378
Lihat kuota layanan EKS di AWS Management Console	1378
Lihat kuota layanan EKS dengan CLI AWS	1379
Amazon EKS service quotas	1380
AWS Kuota layanan Fargate	1380
Keamanan	1382
Praktik terbaik	1383
Menganalisis kerentanan	1383
Patokan Pusat Keamanan Internet (CIS) untuk Amazon EKS	1383
Amazon EKS versi platform	1384
Daftar kerentanan sistem operasi	1384
Deteksi node dengan Amazon Inspector	1385
Deteksi cluster dan node dengan Amazon GuardDuty	1385
Validasi kepatuhan	1385
Pertimbangan untuk EKS	1386
Keamanan infrastruktur	1386
Ketahanan	1391
Pertimbangan untuk Kubernetes	1392
Penandatanganan sertifikat	1392
Peran dan pengguna default	1395
Aktifkan enkripsi rahasia	1400
AWS Secrets Manager	1405
Enkripsi amplop default untuk semua Data API Kubernetes	1405

Pertimbangan untuk EKS Auto	1413
Keamanan dan otentikasi API	1414
Keamanan jaringan	1414
EC2 keamanan instance terkelola	1415
Manajemen sumber daya otomatis	1416
Praktik terbaik keamanan	1417
Referensi IAM	1417
Audiens	1417
Mengautentikasi dengan identitas	1418
Mengelola akses menggunakan kebijakan	1421
Amazon EKS dan IAM	1424
Kebijakan berbasis identitas	1428
Peran terkait layanan	1436
Peran IAM eksekusi pod	1453
Peran konektor IAM	1458
AWS kebijakan terkelola	1462
Pemecahan Masalah	1489
IAM role kluster	1491
IAM role simpul	1495
Mode Otomatis kluster peran IAM	1500
Peran IAM simpul Mode Otomatis	1504
Monitor cluster	1508
Pemantauan dan pencatatan di Amazon EKS	1508
Alat pemantauan dan pencatatan Amazon EKS	1510
Dasbor observabilitas	1514
Ringkasan	1514
Kesehatan cluster	1514
Kontrol pemantauan pesawat	1514
Wawasan cluster	1517
Masalah kesehatan simpul	1517
Metrik Prometheus	1517
Langkah 1: Aktifkan metrik Prometheus	1518
Langkah 2: Gunakan metrik Prometheus	1520
Langkah 3: Kelola pencakar Prometheus	1520
Terapkan menggunakan Helm	1521
Bidang kontrol	1524

Amazon CloudWatch	1532
Metrik dasar di Amazon CloudWatch	1532
Operator CloudWatch Observabilitas Amazon	1541
Kontrol log pesawat	1542
Mengaktifkan atau menonaktifkan log bidang kontrol	1543
Lihat log bidang kontrol kluster	1545
AWS CloudTrail	1547
Referensi	1547
Entri berkas log	1548
Metrik grup Auto Scaling	1551
Operator ADOT	1551
Dasbor Amazon EKS	1552
Apa itu Dasbor Amazon EKS?	1552
Bagaimana dasbor menggunakan AWS Organizations?	1552
Aktifkan Dasbor EKS menggunakan AWS konsol	1554
Lihat dasbor EKS	1555
Konfigurasikan dasbor	1555
Nonaktifkan dasbor EKS menggunakan AWS konsol	1556
Memecahkan masalah dasbor EKS	1557
Konfigurasikan organisasi	1557
Bekerja dengan layanan yang lain	1564
AWS CloudFormation	1564
Amazon EKS dan AWS CloudFormation template	1564
Pelajari lebih lanjut tentang AWS CloudFormation	1565
Amazon Detective	1565
Gunakan Amazon Detective dengan Amazon EKS	1566
Amazon GuardDuty	1566
AWS Hub Ketahanan	1568
Amazon Security Lake	1568
Manfaat menggunakan Security Lake dengan Amazon Amazon EKS	1568
Mengaktifkan Danau Keamanan untuk Amazon EKS	1569
Menganalisis Log EKS di Danau Keamanan	1569
Kisi VPC Amazon	1570
AWS Local Zones	1570
Pemecahan Masalah	1572
Kapasitas tidak mencukupi	1572

Simpul gagal untuk bergabung dengan klaster	1572
Tidak sah atau akses ditolak (kubectl)	1574
hostname doesn't match	1575
getsockopt: no route to host	1575
Instances failed to join the Kubernetes cluster	1576
Kode kesalahan grup node terkelola	1576
Not authorized for images	1581
Node dalam NotReady keadaan	1581
EKS Log Kolektor	1582
Jaringan waktu aktif kontainer belum siap	1583
Waktu habis handshake TLS	1584
InvalidClientTokenId	1585
Grup node harus cocok dengan versi Kubernetes sebelum memutakhirkan bidang kontrol	1585
Saat meluncurkan banyak node, ada Too Many Requests kesalahan	1586
Respons kesalahan HTTP 401 tidak sah pada permintaan server API Kubernetes	1586
Versi platform Amazon EKS lebih dari dua versi di belakang versi platform saat ini	1587
Kesehatan cluster FAQs dan kode kesalahan dengan jalur resolusi	1590
Konektor Amazon EKS	1597
Pertimbangan Amazon EKS Connector	1597
Peran IAM yang diperlukan untuk Konektor Amazon EKS	1598
Connect sebuah cluster	1598
Pertimbangan	1599
Prasyarat	1599
Langkah 1: Mendaftarkan cluster	1599
Langkah 2: Memasang eks-connector agen	1602
Langkah selanjutnya	1604
Berikan akses ke cluster	1604
Prasyarat	1604
Deregister sebuah cluster	1606
Deregister cluster Kubernetes	1606
Bersihkan sumber daya di klaster Kubernetes	1607
Memecahkan Masalah Konektor EKS	1608
Pemecahan masalah dasar	1608
Masalah helm: 403 Terlarang	1610
Kesalahan konsol: cluster macet dalam status Tertunda	1610

Kesalahan konsol: Sistem pengguna: serviceaccount:eks-connector:eks-connector tidak dapat menyamar sebagai pengguna sumber daya dalam grup API pada cakupan cluster ..	1611
Kesalahan konsol: [...] dilarang: Pengguna [...] tidak dapat mencantumkan sumber daya [...] dalam grup API di cakupan cluster	1611
Kesalahan konsol: Amazon EKS tidak dapat berkomunikasi dengan server API cluster Kubernetes Anda. Cluster harus dalam keadaan AKTIF agar koneksi berhasil. Coba lagi dalam beberapa menit.	1612
Pod konektor Amazon EKS sedang crash looping	1612
Gagal memulai eks-konektor: InvalidActivation	1613
Node cluster tidak memiliki konektivitas keluar	1614
Pod konektor Amazon EKS sedang dalam ImagePullBackOff keadaan	1614
Pertanyaan umum	1615
Pertimbangan keamanan	1616
AWS tanggung jawab	1616
Tanggung jawab pelanggan	1616
Amazon EKS di AWS Outposts	1618
Kapan menggunakan setiap opsi penerapan	1618
Membandingkan opsi penerapan	1619
Jalankan cluster lokal	1621
AWS Wilayah yang Didukung	1622
Menyebarkan kluster lokal	1622
Versi platform EKS	1633
Buat VPC dan subnet	1646
Bersiaplah untuk terputus	1649
Pertimbangan kapasitas	1654
Memecahkan masalah cluster	1657
Simpul	1668
eksctl	1670
AWS Management Console	1671
Machine Learning di EKS	1678
Mengapa Memilih EKS untuk AI/ML?	1678
Kasus penggunaan kunci	1678
Studi kasus	1680
Mulai menggunakan Machine Learning di EKS	1680
Memulai dengan ML	1680
Melanjutkan dengan ML di EKS	1682

Siapkan untuk ML	1682
Jalankan Linux GPU AMIs	1683
Jalankan Windows GPU AMIs	1685
Reservasi GPUs untuk MNG	1692
Cadangan GPUs untuk SMN	1695
Node GPU noda	1698
Cluster pelatihan dengan EFA	1700
Siapkan cluster Inferentia	1711
Coba tutorial untuk ML di EKS	1717
Bangun platform AI generatif di EKS	1717
Jalankan kerangka kerja AI generatif khusus di EKS	1718
Maksimalkan kinerja GPU NVIDIA untuk ML di EKS	1718
Jalankan beban kerja pengkodean video di EKS	1718
Mempercepat pemuatan gambar untuk beban kerja inferensi	1718
Memantau beban kerja ML	1718
Pengumuman untuk ML di EKS	1719
Proyek yang terkait dengan Amazon EKS	1720
Support untuk perangkat lunak yang digunakan untuk EKS	1720
Alat manajemen	1721
eksctl	1721
AWS pengendali untuk Kubernetes	1721
Flux CD	1721
CDK untuk Kubernetes	1722
Jaringan	1722
Plugin Amazon VPC CNI untuk Kubernetes	1722
AWS Load Balancer Controller untuk Kubernetes	1722
ExternalDNS	1723
Machine learning	1723
Kubeflow	1723
Auto Scaling	1723
Autoscaler klaster	1723
Karpenter	1723
Eskalator	1724
Pemantauan	1724
Prometheus	1724
Integrasi berkelanjutan/deployment berkesinambungan	1724

Jenkins X	1724
Fitur dan peta jalan baru	1725
Riwayat dokumen	1726
Berkontribusi	1783
Edit satu halaman	1783
Prasyarat	49
Prosedur	102
Ikhtisar permintaan tarik	1785
Edit file dengan GitHub	1786
Prasyarat	49
Prosedur	102
Lihat umpan balik gaya	1787
Instal Vale	1788
Instal ekstensi VS Code Vale	1788
Sinkronisasi Vale	1789
Lihat umpan balik gaya di VS Code	1789
Buat halaman	1789
Buat halaman	1789
Tambahkan halaman ke navigasi	1790
Sisipkan tautan	1790
Tautan ke halaman atau bagian dalam Panduan Pengguna EKS	1791
Tautan ke panduan lain di AWS Dokumen	1791
Tautan ke halaman web eksternal	1791
Buat dengan Amazon Q	1792
Instal Amazon Q dengan VS Code	1792
Masuk ke Amazon Q	1793
Gunakan Amazon Q untuk membuat konten	1793
Kiat	1793
Lihat pratinjau PR	1794
Lihat pratinjau	1795
Batasan pratinjau	1795
AsciiDoc sintaks	1796
Halaman baru	1796
Judul	1796
Pemformatan teks	1796
Daftar	1796

Tautan 1797

Contoh kode 1797

Citra 1798

Tabel 1798

Callout 1798

Termasuk file lainnya 1799

Atribut (mirip dengan entitas) 1799

Prosedur 1799

Bantu tingkatkan halaman ini

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.

Untuk berkontribusi pada panduan pengguna ini, pilih Edit halaman ini pada GitHub tautan yang terletak di panel kanan setiap halaman.

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.

Apa itu Amazon EKS?

Amazon EKS: Manajemen Kubernetes Sederhana

Amazon Elastic Kubernetes Service (EKS) menyediakan layanan Kubernetes yang dikelola sepenuhnya yang menghilangkan kompleksitas pengoperasian kluster Kubernetes. Dengan EKS, Anda dapat:

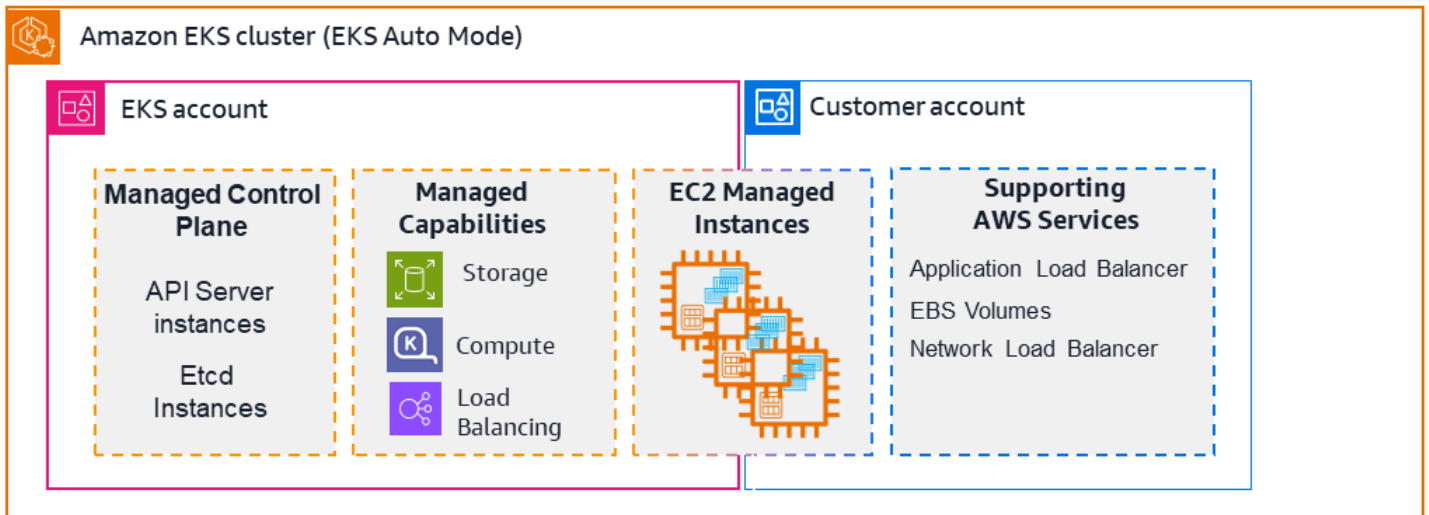
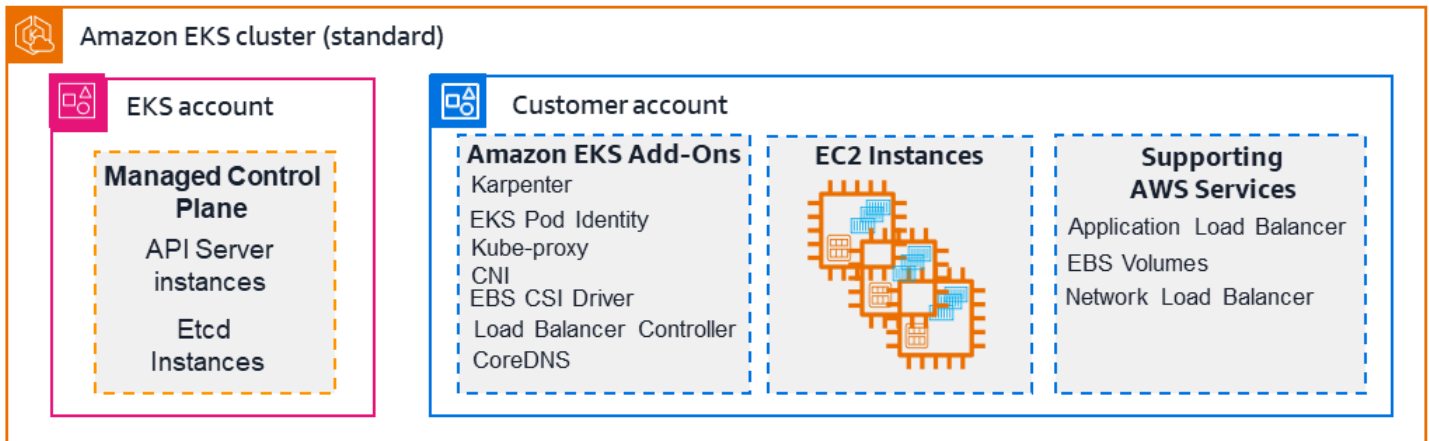
- Menyebarkan aplikasi lebih cepat dengan overhead operasional yang lebih sedikit
- Skala dengan mulus untuk memenuhi tuntutan beban kerja yang berubah
- Meningkatkan keamanan melalui AWS integrasi dan pembaruan otomatis
- Pilih antara EKS standar atau Mode Otomatis EKS otomatis sepenuhnya

[Amazon Elastic Kubernetes Service \(Amazon EKS\)](#) adalah platform perdana untuk menjalankan cluster Kubernetes, baik di cloud Amazon Web Services () maupun di pusat data Anda sendiri (EKS Anywhere dan AWS Amazon EKS Hybrid Nodes).

Amazon EKS menyederhanakan membangun, mengamankan, dan memelihara kluster Kubernetes. Ini bisa lebih hemat biaya dalam menyediakan sumber daya yang cukup untuk memenuhi permintaan puncak daripada memelihara pusat data Anda sendiri. Dua pendekatan utama untuk menggunakan Amazon EKS adalah sebagai berikut:

- Standar EKS: AWS mengelola [bidang kontrol Kubernetes](#) saat Anda membuat cluster dengan EKS. Komponen yang mengelola node, menjadwalkan beban kerja, terintegrasi dengan AWS cloud, dan menyimpan serta menskalakan informasi bidang kontrol untuk menjaga kluster Anda tetap aktif dan berjalan, ditangani untuk Anda secara otomatis.
- EKS Auto Mode: Menggunakan fitur [EKS Auto Mode](#), EKS memperluas kontrolnya untuk mengelola [Node](#) (Kubernetes data plane) juga. Ini menyederhanakan pengelolaan Kubernetes dengan menyediakan infrastruktur secara otomatis, memilih instans komputasi yang optimal, menskalakan sumber daya secara dinamis, terus mengoptimalkan biaya, menambal sistem operasi, dan mengintegrasikan dengan layanan keamanan. AWS

Diagram berikut menggambarkan bagaimana Amazon EKS mengintegrasikan cluster Kubernetes Anda dengan AWS cloud, tergantung pada metode pembuatan kluster yang Anda pilih:



Amazon EKS membantu Anda mempercepat waktu produksi, meningkatkan kinerja, ketersediaan, dan ketahanan, serta meningkatkan keamanan sistem. Untuk informasi selengkapnya, lihat [Amazon Elastic Kubernetes Service](#).

Fitur Amazon EKS

Amazon EKS menyediakan fitur tingkat tinggi berikut:

Antarmuka manajemen

EKS menawarkan beberapa antarmuka untuk menyediakan, mengelola, dan memelihara cluster, termasuk, AWS Management Console Amazon EKS API/, SDKs CDK, CLI, eksctl AWS CLI,, dan Terraform. Untuk informasi selengkapnya, lihat [Memulai](#) dan [Klaster](#).

Alat kontrol akses

EKS mengandalkan fitur Kubernetes dan AWS Identity and Access Management (AWS IAM) untuk [mengelola akses](#) dari pengguna dan beban kerja. Untuk informasi selengkapnya, lihat [the section called “Akses API Kubernetes”](#) dan [the section called “Akses beban kerja ke AWS”](#).

Menghitung sumber daya

Untuk [sumber daya komputasi](#), EKS memungkinkan berbagai jenis EC2 instans Amazon dan AWS inovasi seperti Nitro dan Graviton dengan Amazon EKS agar Anda dapat mengoptimalkan komputasi untuk beban kerja Anda. Untuk informasi selengkapnya, lihat [Kelola komputasi](#).

Penyimpanan

Mode Otomatis EKS secara otomatis membuat kelas penyimpanan menggunakan [volume EBS](#). Menggunakan driver Container Storage Interface (CSI), Anda juga dapat menggunakan Amazon S3, Amazon EFS, Amazon FSX, dan Amazon File Cache untuk kebutuhan penyimpanan aplikasi Anda. Untuk informasi selengkapnya, lihat [Penyimpanan data aplikasi](#).

Keamanan

Model tanggung jawab bersama digunakan karena berkaitan dengan [Keamanan di Amazon EKS](#). Untuk informasi selengkapnya, lihat [Praktik terbaik keamanan](#), [Keamanan infrastruktur](#), dan [keamanan Kubernetes](#).

Alat pemantauan

Gunakan [dasbor observabilitas](#) untuk memantau kluster Amazon EKS. [Alat pemantauan termasuk Prometheus,, Cloudtrail CloudWatch, dan Operator ADOT](#). Untuk informasi selengkapnya tentang dasbor, server metrik, dan alat lainnya, lihat [biaya kluster EKS](#) dan Server Metrik [Kubernetes](#).

Kompatibilitas dan dukungan Kubernetes

Amazon EKS bersertifikat Kubernetes-Conformant, sehingga Anda dapat menerapkan aplikasi yang kompatibel dengan Kubernetes tanpa refactoring dan menggunakan tooling dan plugin komunitas Kubernetes. EKS menawarkan [dukungan standar dan dukungan tambahan](#) untuk Kubernetes. Untuk informasi selengkapnya, lihat [the section called “Versi Kubernetes”](#).

Layanan terkait

Layanan untuk digunakan dengan Amazon EKS

Anda dapat menggunakan AWS layanan lain dengan kluster yang Anda gunakan menggunakan Amazon EKS:

Amazon EC2

[Dapatkan kapasitas komputasi sesuai permintaan dan skalabel dengan Amazon. EC2](#)

Amazon EBS

Lampirkan sumber daya penyimpanan blok berkinerja tinggi yang dapat diskalakan dengan [Amazon EBS](#).

Amazon ECR

Simpan gambar kontainer dengan aman dengan [Amazon ECR](#).

Amazon CloudWatch

Pantau AWS sumber daya dan aplikasi secara real time dengan [Amazon CloudWatch](#).

Amazon Prometheus

Lacak metrik untuk aplikasi kontainer dengan [Amazon Managed Service untuk Prometheus](#).

Elastic Load Balancing

Mendistribusikan lalu lintas masuk ke beberapa target dengan [Elastic Load Balancing](#).

Amazon GuardDuty

Mendeteksi ancaman ke kluster EKS dengan [Amazon GuardDuty](#).

AWS Hub Ketahanan

Nilai ketahanan kluster EKS dengan [AWS Resilience Hub](#).

Harga Amazon EKS

Amazon EKS memiliki harga per cluster berdasarkan dukungan versi cluster Kubernetes, harga untuk Amazon EKS Auto Mode, dan harga per vCPU untuk Amazon EKS Hybrid Nodes.

Saat menggunakan Amazon EKS, Anda membayar secara terpisah untuk AWS sumber daya yang Anda gunakan untuk menjalankan aplikasi Anda di node pekerja Kubernetes. Misalnya, jika Anda menjalankan node pekerja Kubernetes sebagai EC2 instans Amazon dengan volume Amazon

EBS dan alamat IPv4 publik, Anda akan dikenakan biaya untuk kapasitas instans melalui Amazon, kapasitas volume melalui EC2 Amazon EBS, dan alamat IPv4 melalui Amazon VPC.

Kunjungi halaman harga masing-masing AWS layanan yang Anda gunakan dengan aplikasi Kubernetes Anda untuk informasi harga terperinci.

- Untuk harga kluster Amazon EKS, Mode Otomatis Amazon EKS, dan Amazon EKS Hybrid Nodes, lihat [Harga Amazon EKS](#).
- Untuk EC2 harga Amazon, lihat Harga [EC2 Sesuai Permintaan Amazon dan Harga EC2 Spot Amazon](#).
- [Untuk harga AWS Fargate, lihat Harga Fargate AWS](#).
- Anda dapat menggunakan paket tabungan Anda untuk komputasi yang digunakan di kluster Amazon EKS. Untuk informasi lebih lanjut, lihat: [Harga dengan Savings Plans](#).

Kasus penggunaan umum di Amazon EKS

Amazon EKS menawarkan layanan Kubernetes terkelola yang kuat AWS, yang dirancang untuk mengoptimalkan aplikasi kontainer. Berikut ini adalah beberapa kasus penggunaan Amazon EKS yang paling umum, membantu Anda memanfaatkan kekuatannya untuk kebutuhan spesifik Anda.

Menyebarkan aplikasi ketersediaan tinggi

Menggunakan [Elastic Load Balancing](#), Anda dapat memastikan bahwa aplikasi Anda sangat tersedia di beberapa [Availability Zone](#).

Membangun arsitektur microservices

Gunakan fitur penemuan layanan Kubernetes dengan [AWS Cloud Map](#) atau Amazon [VPC Lattice](#) untuk membangun sistem yang tangguh.

Mengotomatiskan proses rilis perangkat lunak

Kelola pipeline continuous integration and continuous deployment (CI/CD) yang menyederhanakan proses pembuatan, pengujian, dan penyebaran aplikasi secara otomatis.

Menjalankan aplikasi tanpa server

Gunakan [AWS Fargate](#) dengan Amazon EKS untuk menjalankan aplikasi tanpa server. Ini berarti Anda hanya dapat fokus pada pengembangan aplikasi, sementara Amazon EKS dan Fargate menangani infrastruktur yang mendasarinya.

Menjalankan beban kerja pembelajaran mesin

Amazon EKS kompatibel dengan kerangka kerja pembelajaran mesin populer seperti [TensorFlow](#), [MXNet](#), dan [PyTorch](#). Dengan dukungan GPU, Anda dapat menangani tugas pembelajaran mesin yang kompleks secara efektif.

Menerapkan secara konsisten di tempat dan di cloud

Untuk menyederhanakan menjalankan Kubernetes di lingkungan lokal, Anda dapat menggunakan kluster, fitur, dan alat Amazon EKS yang sama untuk menjalankan node yang dikelola sendiri di Outposts AWS atau dapat menggunakan Amazon EKS Hybrid Nodes dengan infrastruktur Anda sendiri. Untuk lingkungan mandiri dengan celah udara, Anda dapat menggunakan [Amazon EKS Anywhere](#) untuk mengotomatiskan manajemen siklus hidup kluster Kubernetes di infrastruktur Anda sendiri.

Menjalankan pemrosesan batch yang hemat biaya dan beban kerja data besar

Manfaatkan [Instans Spot](#) untuk menjalankan pemrosesan batch dan beban kerja data besar seperti [Apache Hadoop](#) dan [Spark](#), dengan biaya yang lebih murah. Ini memungkinkan Anda memanfaatkan EC2 kapasitas Amazon yang tidak terpakai dengan harga diskon.

Mengamankan aplikasi dan memastikan kepatuhan

Menerapkan praktik keamanan yang kuat dan menjaga kepatuhan terhadap Amazon EKS, yang terintegrasi dengan layanan AWS keamanan seperti [AWS Identity and Access Management](#) (IAM), [Amazon Virtual Private Cloud](#) (Amazon VPC), dan [AWS Key Management Service](#) (KMS). AWS Ini memastikan privasi dan perlindungan data sesuai standar industri.

Arsitektur Amazon EKS

Amazon EKS selaras dengan arsitektur cluster umum Kubernetes. Untuk informasi selengkapnya, lihat [Komponen Kubernetes](#) di dokumentasi Kubernetes. Bagian berikut merangkum beberapa detail arsitektur tambahan untuk Amazon EKS.

Bidang kontrol

Amazon EKS memastikan setiap cluster memiliki bidang kontrol Kubernetes yang unik. Desain ini membuat infrastruktur masing-masing cluster terpisah, tanpa tumpang tindih antara cluster atau akun. AWS Pengaturan meliputi:

Komponen terdistribusi

Bidang kontrol memposisikan setidaknya dua instance server API dan tiga instans [etcd](#) di tiga AWS Availability Zone dalam suatu Region. AWS

Kinerja optimal

Amazon EKS secara aktif memonitor dan menyesuaikan instans bidang kontrol untuk mempertahankan kinerja puncak.

Ketahanan

Jika instance bidang kontrol terputus-putus, Amazon EKS dengan cepat menggantikannya, menggunakan Availability Zone yang berbeda jika diperlukan.

Waktu aktif yang konsisten

Dengan menjalankan klaster di beberapa Availability Zone, [ketersediaan titik akhir server API yang andal Service Level Agreement \(SLA\)](#) tercapai.

Amazon EKS menggunakan Amazon Virtual Private Cloud (Amazon VPC) untuk membatasi lalu lintas antara komponen bidang kontrol dalam satu cluster. Komponen klaster tidak dapat melihat atau menerima komunikasi dari klaster atau AWS akun lain, kecuali jika diizinkan oleh kebijakan kontrol akses berbasis peran (RBAC) Kubernetes.

Hitung

Selain bidang kontrol, cluster Amazon EKS memiliki satu set mesin pekerja yang disebut node. Memilih jenis node cluster Amazon EKS yang sesuai sangat penting untuk memenuhi persyaratan spesifik Anda dan mengoptimalkan pemanfaatan sumber daya. Amazon EKS menawarkan tipe simpul utama berikut:

Mode Otomatis EKS

[Mode Otomatis EKS](#) memperluas AWS manajemen di luar bidang kontrol untuk menyertakan bidang data, mengotomatisasi manajemen infrastruktur klaster. Ini mengintegrasikan kemampuan inti Kubernetes sebagai komponen bawaan, termasuk komputasi autoscaling, jaringan, load balancing, DNS, penyimpanan, dan dukungan GPU. Mode Otomatis EKS mengelola node secara dinamis berdasarkan tuntutan beban kerja, menggunakan immutable AMIs dengan fitur keamanan yang disempurnakan. Ini mengotomatiskan pembaruan dan peningkatan sambil menghormati Anggaran Gangguan Pod, dan menyertakan komponen terkelola yang jika tidak

memerlukan manajemen add-on. Opsi ini sangat ideal bagi pengguna yang ingin memanfaatkan AWS keahlian untuk day-to-day operasi, meminimalkan overhead operasional, dan fokus pada pengembangan aplikasi daripada manajemen infrastruktur.

AWS Fargate

[Fargate](#) adalah mesin komputasi tanpa server untuk kontainer yang menghilangkan kebutuhan untuk mengelola instance yang mendasarinya. Dengan Fargate, Anda menentukan kebutuhan sumber daya aplikasi Anda, dan AWS secara otomatis menyediakan, menskalakan, dan memelihara infrastruktur. Opsi ini sangat ideal untuk pengguna yang memprioritaskan ease-of-use dan ingin berkonsentrasi pada pengembangan dan penyebaran aplikasi daripada mengelola infrastruktur.

Karpenter

[Karpenter](#) adalah autoscaler cluster Kubernetes yang fleksibel dan berkinerja tinggi yang membantu meningkatkan ketersediaan aplikasi dan efisiensi klaster. Karpenter meluncurkan sumber daya komputasi berukuran tepat sebagai respons terhadap perubahan beban aplikasi. Opsi ini dapat menyediakan sumber daya just-in-time komputasi yang memenuhi persyaratan beban kerja Anda.

Grup simpul terkelola

[Grup node terkelola](#) adalah perpaduan antara otomatisasi dan kustomisasi untuk mengelola kumpulan EC2 instans Amazon dalam klaster Amazon EKS. AWS menangani tugas-tugas seperti menambal, memperbarui, dan menskalakan node, mengurangi aspek operasional. Secara paralel, kubelet argumen kustom didukung, membuka kemungkinan untuk kebijakan manajemen CPU dan memori tingkat lanjut. Selain itu, mereka meningkatkan keamanan melalui peran AWS Identity and Access Management (IAM) and Access Management (IAM) untuk akun layanan, sambil membatasi kebutuhan akan izin terpisah per cluster.

Node yang dikelola sendiri

[Node yang dikelola sendiri](#) menawarkan kontrol penuh atas EC2 instans Amazon Anda dalam klaster Amazon EKS. Anda bertanggung jawab untuk mengelola, menskalakan, dan memelihara node, memberi Anda kendali penuh atas infrastruktur yang mendasarinya. Opsi ini cocok untuk pengguna yang membutuhkan kontrol granular dan penyesuaian node mereka dan siap menginvestasikan waktu dalam mengelola dan memelihara infrastruktur mereka.

Node Hibrida Amazon EKS

Dengan [Amazon EKS Hybrid Nodes](#), Anda dapat menggunakan infrastruktur lokal dan edge sebagai node di kluster Amazon EKS. Amazon EKS Hybrid Nodes menyatukan manajemen

Kubernetes di seluruh lingkungan dan menurunkan manajemen bidang kontrol Kubernetes untuk aplikasi lokal dan edge Anda. AWS

Konsep Kubernetes

[Amazon Elastic Kubernetes Service \(Amazon EKS\) AWS adalah layanan terkelola berdasarkan proyek Kubernetes open source.](#) Meskipun ada hal-hal yang perlu Anda ketahui tentang bagaimana layanan Amazon EKS terintegrasi dengan AWS Cloud (terutama ketika Anda pertama kali membuat kluster Amazon EKS), setelah aktif dan berjalan, Anda menggunakan cluster Amazon EKS Anda dengan cara yang sama seperti yang Anda lakukan pada cluster Kubernetes lainnya. Jadi untuk mulai mengelola kluster Kubernetes dan menerapkan beban kerja, Anda memerlukan setidaknya pemahaman dasar tentang konsep Kubernetes.

Halaman ini membagi konsep Kubernetes menjadi tiga bagian: [the section called “Mengapa Kubernetes?”](#), dan [the section called “Kluster”](#) [the section called “Beban kerja”](#) Bagian pertama menjelaskan nilai menjalankan layanan Kubernetes, khususnya sebagai layanan terkelola seperti Amazon EKS. Bagian Beban Kerja mencakup bagaimana aplikasi Kubernetes dibangun, disimpan, dijalankan, dan dikelola. Bagian Cluster menjabarkan berbagai komponen yang membentuk kluster Kubernetes dan apa tanggung jawab Anda untuk membuat dan memelihara kluster Kubernetes.

Topik

- [Mengapa Kubernetes?](#)
- [Kluster](#)
- [Beban kerja](#)
- [Langkah selanjutnya](#)

Saat Anda membaca konten ini, tautan akan mengarahkan Anda ke deskripsi lebih lanjut tentang konsep Kubernetes dalam dokumentasi Amazon EKS dan Kubernetes, jika Anda ingin menyelami topik apa pun yang kami bahas di sini. Untuk detail tentang cara Amazon EKS mengimplementasikan bidang kontrol Kubernetes dan fitur komputasi, lihat [the section called “Arsitektur”](#)

Mengapa Kubernetes?

Kubernetes dirancang untuk meningkatkan ketersediaan dan skalabilitas saat menjalankan aplikasi kontainer berkualitas produksi yang kritis terhadap misi. Daripada hanya menjalankan Kubernetes pada satu mesin (meskipun itu mungkin), Kubernetes mencapai tujuan tersebut dengan

memungkinkan Anda menjalankan aplikasi di seluruh rangkaian komputer yang dapat memperluas atau mengontrak untuk memenuhi permintaan. Kubernetes menyertakan fitur-fitur yang memudahkan Anda untuk:

- Menerapkan aplikasi pada beberapa mesin (menggunakan kontainer yang di-deploy di Pod)
- Pantau kesehatan kontainer dan mulai ulang kontainer yang gagal
- Skalikan kontainer ke atas dan ke bawah berdasarkan beban
- Perbarui wadah dengan versi baru
- Alokasikan sumber daya antar kontainer
- Menyeimbangkan lalu lintas di seluruh mesin

Memiliki Kubernetes mengotomatiskan jenis tugas kompleks ini memungkinkan pengembang aplikasi untuk fokus membangun dan meningkatkan beban kerja aplikasi mereka, daripada mengkhawatirkan infrastruktur. Pengembang biasanya membuat file konfigurasi, diformat sebagai file YAML, yang menggambarkan keadaan aplikasi yang diinginkan. Ini dapat mencakup kontainer mana yang akan dijalankan, batas sumber daya, jumlah replika Pod, alokasi CPU/memori, aturan afinitas, dan banyak lagi.

Atribut Kubernetes

Untuk mencapai tujuannya, Kubernetes memiliki atribut berikut:

- Containerized - Kubernetes adalah alat orkestrasi kontainer. Untuk menggunakan Kubernetes, Anda harus terlebih dahulu memiliki aplikasi Anda dalam kontainerisasi. Tergantung pada jenis aplikasi, ini bisa sebagai satu set layanan mikro, sebagai pekerjaan batch atau dalam bentuk lain. [Kemudian, aplikasi Anda dapat memanfaatkan alur kerja Kubernetes yang mencakup ekosistem alat yang sangat besar, di mana kontainer dapat disimpan sebagai gambar dalam registri kontainer, di-deploy ke cluster Kubernetes, dan dijalankan pada node yang tersedia.](#) Anda dapat membangun dan menguji kontainer individual di komputer lokal Anda dengan Docker atau [runtime kontainer](#) lain, sebelum menerapkannya ke cluster Kubernetes Anda.
- Scalable — Jika permintaan untuk aplikasi Anda melebihi kapasitas instans yang sedang berjalan dari aplikasi tersebut, Kubernetes dapat meningkatkan skala. Sesuai kebutuhan, Kubernetes dapat mengetahui apakah aplikasi membutuhkan lebih banyak CPU atau memori dan merespons dengan memperluas kapasitas yang tersedia secara otomatis atau menggunakan lebih banyak kapasitas yang ada. [Penskalaan dapat dilakukan pada level Pod, jika ada cukup komputasi yang tersedia untuk menjalankan lebih banyak instance aplikasi \(horizontal Pod autoscaling\), atau pada](#)

[tingkat node, jika lebih banyak node perlu dibawa untuk menangani peningkatan kapasitas \(Cluster Autoscaler atau Karpenter\)](#). Karena kapasitas tidak lagi diperlukan, layanan ini dapat menghapus Pod yang tidak perlu dan mematikan node yang tidak dibutuhkan.

- **Available** — Jika aplikasi atau node menjadi tidak sehat atau tidak tersedia, Kubernetes dapat memindahkan beban kerja yang sedang berjalan ke node lain yang tersedia. Anda dapat memaksakan masalah hanya dengan menghapus instance yang sedang berjalan dari beban kerja atau node yang menjalankan beban kerja Anda. Intinya di sini adalah bahwa beban kerja dapat diangkat di lokasi lain jika mereka tidak dapat lagi berjalan di tempat mereka berada.
- **Deklaratif** — Kubernetes menggunakan rekonsiliasi aktif untuk terus-menerus memeriksa apakah status yang Anda deklarasikan untuk kluster Anda cocok dengan status sebenarnya. Dengan menerapkan [objek Kubernetes](#) ke cluster, biasanya melalui file konfigurasi berformat YAML, Anda dapat, misalnya, meminta untuk memulai beban kerja yang ingin Anda jalankan di cluster Anda. Anda nantinya dapat mengubah konfigurasi untuk melakukan sesuatu seperti menggunakan versi kontainer yang lebih baru atau mengalokasikan lebih banyak memori. Kubernetes akan melakukan apa yang perlu dilakukan untuk menetapkan keadaan yang diinginkan. Ini dapat mencakup membawa node ke atas atau ke bawah, menghentikan dan memulai ulang beban kerja, atau menarik kontainer yang diperbarui.
- **Composable** — Karena aplikasi biasanya terdiri dari beberapa komponen, Anda ingin dapat mengelola satu set komponen ini (sering diwakili oleh beberapa kontainer) bersama-sama. Sementara Docker Compose menawarkan cara untuk melakukan ini secara langsung dengan Docker, perintah Kubernetes [Kompose](#) dapat membantu Anda melakukannya dengan Kubernetes. Lihat [Translate a Docker Compose File ke Kubernetes Resources](#) untuk contoh cara melakukannya.
- **Extensible** — Tidak seperti perangkat lunak berpemilik, proyek Kubernetes open source dirancang untuk terbuka bagi Anda memperluas Kubernetes dengan cara apa pun yang Anda inginkan untuk memenuhi kebutuhan Anda. APIs dan file konfigurasi terbuka untuk modifikasi langsung. Pihak ketiga didorong untuk menulis [Controller](#) mereka sendiri, untuk memperluas infrastruktur dan fitur Kubernetes pengguna akhir. [Webhook](#) memungkinkan Anda menyiapkan aturan kluster untuk menegakkan kebijakan dan beradaptasi dengan perubahan kondisi. [Untuk lebih banyak ide tentang cara memperluas kluster Kubernetes, lihat Memperluas Kubernetes.](#)
- **Portable** — Banyak organisasi telah menstandarisasi operasi mereka di Kubernetes karena memungkinkan mereka untuk mengelola semua kebutuhan aplikasi mereka dengan cara yang sama. Pengembang dapat menggunakan pipeline yang sama untuk membangun dan menyimpan aplikasi kontainer. Aplikasi tersebut kemudian dapat diterapkan ke kluster Kubernetes yang berjalan di lokasi, di cloud, di point-of-sales terminal di restoran, atau pada perangkat IOT yang

tersebar di seluruh situs jarak jauh perusahaan. Sifat open source-nya memungkinkan orang untuk mengembangkan distribusi Kubernetes khusus ini, bersama dengan alat yang diperlukan untuk mengelolanya.

Mengelola Kubernetes

Kode sumber Kubernetes tersedia secara bebas, jadi dengan peralatan Anda sendiri, Anda dapat menginstal dan mengelola Kubernetes sendiri. Namun, Kubernetes yang mengelola diri sendiri membutuhkan keahlian operasional yang mendalam dan membutuhkan waktu dan upaya untuk mempertahankannya. Karena alasan tersebut, kebanyakan orang yang menerapkan beban kerja produksi memilih penyedia cloud (seperti Amazon EKS) atau penyedia lokal (seperti Amazon EKS Anywhere) dengan distribusi Kubernetes yang telah diuji sendiri dan dukungan para ahli Kubernetes. Ini memungkinkan Anda untuk menurunkan banyak angkat berat yang tidak berdiferensiasi yang diperlukan untuk mempertahankan cluster Anda, termasuk:

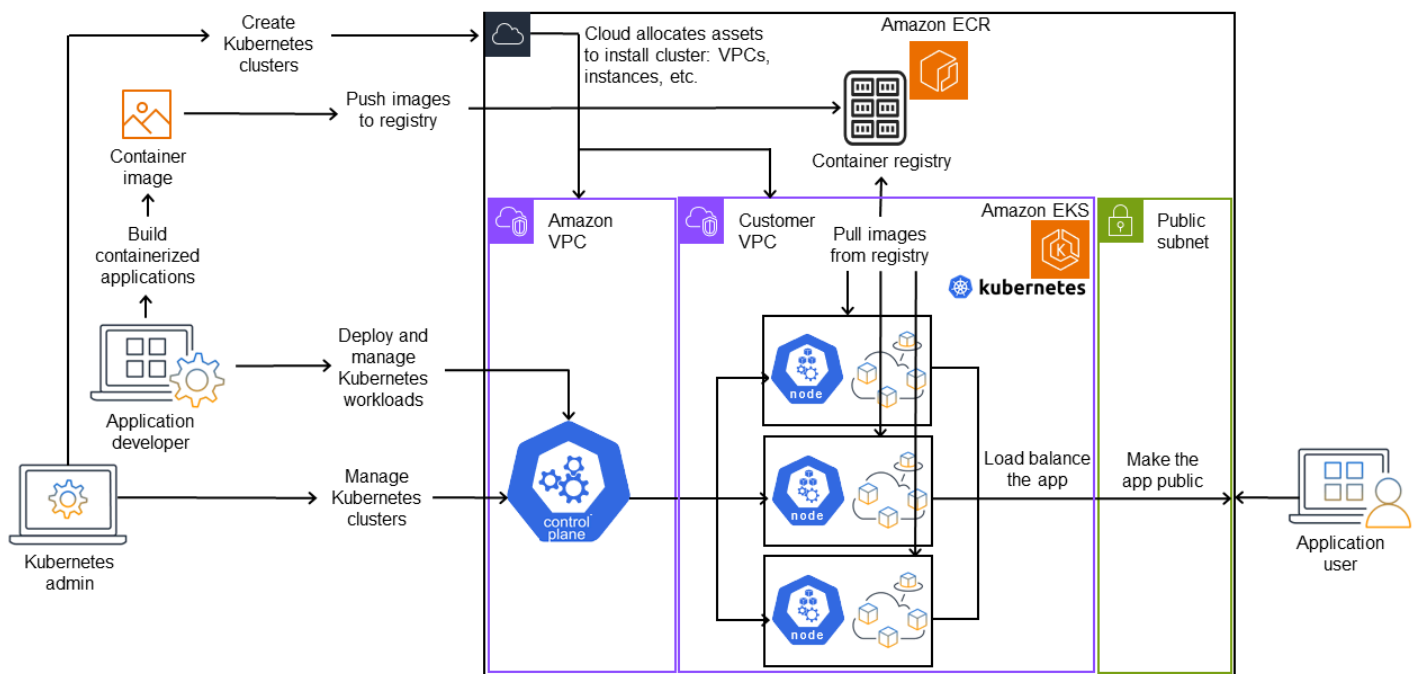
- **Hardware** — Jika Anda tidak memiliki perangkat keras yang tersedia untuk menjalankan Kubernetes sesuai kebutuhan Anda, penyedia cloud seperti AWS Amazon EKS dapat menghemat biaya di muka. Dengan Amazon EKS, ini berarti Anda dapat menggunakan sumber daya cloud terbaik yang ditawarkan oleh AWS, termasuk instans komputer (Amazon Elastic Compute Cloud), lingkungan pribadi Anda sendiri (Amazon VPC), identitas pusat dan manajemen izin (IAM), dan penyimpanan (Amazon EBS). AWS mengelola komputer, jaringan, pusat data, dan semua komponen fisik lainnya yang diperlukan untuk menjalankan Kubernetes. Demikian juga, Anda tidak perlu merencanakan pusat data Anda untuk menangani kapasitas maksimum pada hari-hari permintaan tertinggi Anda. Untuk Amazon EKS Anywhere, atau klaster Kubernetes lainnya di premis, Anda bertanggung jawab untuk mengelola infrastruktur yang digunakan dalam penerapan Kubernetes Anda, tetapi Anda masih dapat mengandalkan AWS untuk membantu Anda memperbarui Kubernetes.
- **Manajemen pesawat kontrol** — Amazon EKS mengelola keamanan dan ketersediaan pesawat kontrol Kubernetes yang AWS di-host, yang bertanggung jawab untuk menjadwalkan kontainer, mengelola ketersediaan aplikasi, dan tugas-tugas utama lainnya, sehingga Anda dapat fokus pada beban kerja aplikasi Anda. Jika cluster Anda rusak, AWS harus memiliki sarana untuk memulihkan cluster Anda ke status berjalan. Untuk Amazon EKS Anywhere, Anda akan mengelola sendiri pesawat kontrol.
- **Upgrade yang diuji** - Saat Anda meng-upgrade klaster, Anda dapat mengandalkan Amazon EKS atau Amazon EKS Anywhere untuk menyediakan versi distribusi Kubernetes mereka yang telah diuji.

- **Add-on** — Ada ratusan proyek yang dibangun untuk memperluas dan bekerja dengan Kubernetes yang dapat Anda tambahkan ke infrastruktur kluster Anda atau gunakan untuk membantu menjalankan beban kerja Anda. Alih-alih membangun dan mengelola add-on itu sendiri, AWS sediakan [the section called “Add-on Amazon EKS”](#) yang dapat Anda gunakan dengan cluster Anda. Amazon EKS Anywhere menyediakan [Paket Terkurasi](#) yang mencakup pembuatan banyak proyek open source populer. Jadi Anda tidak perlu membangun perangkat lunak sendiri atau mengelola patch keamanan kritis, perbaikan bug, atau peningkatan. Demikian juga, jika default memenuhi kebutuhan Anda, biasanya konfigurasi yang sangat sedikit dari add-on tersebut diperlukan. Lihat [the section called “Perluas Cluster”](#) detail tentang memperluas kluster Anda dengan add-on.

Kubernetes beraksi

Diagram berikut menunjukkan aktivitas utama yang akan Anda lakukan sebagai Admin Kubernetes atau Pengembang Aplikasi untuk membuat dan menggunakan kluster Kubernetes. Dalam prosesnya, ini menggambarkan bagaimana komponen Kubernetes berinteraksi satu sama lain, menggunakan AWS cloud sebagai contoh penyedia cloud yang mendasarinya.

A Kubernetes cluster in action



Admin Kubernetes membuat kluster Kubernetes menggunakan alat khusus untuk jenis penyedia tempat kluster akan dibangun. Contoh ini menggunakan AWS cloud sebagai penyedia, yang

menawarkan layanan Kubernetes terkelola yang disebut Amazon EKS. Layanan terkelola secara otomatis mengalokasikan sumber daya yang diperlukan untuk membuat klaster, termasuk membuat dua Virtual Private Clouds VPCs (Amazon) baru untuk klaster, menyiapkan jaringan, dan memetakan izin Kubernetes langsung ke yang baru untuk manajemen aset cloud. VPCs Layanan terkelola juga melihat bahwa layanan pesawat kontrol memiliki tempat untuk dijalankan dan mengalokasikan nol atau lebih EC2 instance Amazon sebagai node Kubernetes untuk menjalankan beban kerja. AWS mengelola satu VPC Amazon itu sendiri untuk bidang kontrol, sedangkan VPC Amazon lainnya berisi node pelanggan yang menjalankan beban kerja.

Banyak tugas Admin Kubernetes ke depan dilakukan dengan menggunakan alat Kubernetes seperti `kubectl`. Alat itu membuat permintaan layanan langsung ke bidang kontrol cluster. Cara kueri dan perubahan dibuat ke cluster kemudian sangat mirip dengan cara Anda melakukannya di klaster Kubernetes mana pun.

Pengembang aplikasi yang ingin menyebarkan beban kerja ke cluster ini dapat melakukan beberapa tugas. Pengembang perlu membangun aplikasi menjadi satu atau beberapa gambar kontainer, lalu mendorong gambar tersebut ke registri kontainer yang dapat diakses oleh cluster Kubernetes. AWS menawarkan Amazon Elastic Container Registry (Amazon ECR) Registry ECR) untuk tujuan itu.

Untuk menjalankan aplikasi, pengembang dapat membuat file konfigurasi berformat YAML yang memberi tahu cluster cara menjalankan aplikasi, termasuk kontainer mana yang akan ditarik dari registri dan cara membungkus kontainer tersebut dalam Pod. Bidang kontrol (scheduler) menjadwalkan kontainer ke satu atau lebih node dan runtime kontainer pada setiap node benar-benar menarik dan menjalankan kontainer yang diperlukan. Pengembang juga dapat mengatur penyeimbang beban aplikasi untuk menyeimbangkan lalu lintas ke kontainer yang tersedia yang berjalan pada setiap node dan mengekspos aplikasi sehingga tersedia di jaringan publik ke dunia luar. Dengan semua itu dilakukan, seseorang yang ingin menggunakan aplikasi dapat terhubung ke titik akhir aplikasi untuk mengaksesnya.

Bagian berikut membahas detail masing-masing fitur ini, dari perspektif Kubernetes Clusters dan Workloads.

Klaster

Jika tugas Anda adalah memulai dan mengelola klaster Kubernetes, Anda harus tahu bagaimana klaster Kubernetes dibuat, ditingkatkan, dikelola, dan dihapus. Anda juga harus tahu komponen apa yang membentuk cluster dan apa yang perlu Anda lakukan untuk mempertahankan komponen tersebut.

Alat untuk mengelola kluster menangani tumpang tindih antara layanan Kubernetes dan penyedia perangkat keras yang mendasarinya. Untuk alasan itu, otomatisasi tugas-tugas ini cenderung dilakukan oleh penyedia Kubernetes (seperti Amazon EKS atau Amazon EKS Anywhere) menggunakan alat yang khusus untuk penyedia. Misalnya, untuk memulai kluster Amazon EKS Anda dapat menggunakan `eksctl create cluster`, sedangkan untuk Amazon EKS Anywhere Anda dapat menggunakan `eksctl anywhere create cluster`. Perhatikan bahwa sementara perintah ini membuat cluster Kubernetes, mereka khusus untuk penyedia dan bukan bagian dari proyek Kubernetes itu sendiri.

Pembuatan kluster dan alat manajemen

Proyek Kubernetes menawarkan alat untuk membuat kluster Kubernetes secara manual. [Jadi jika Anda ingin menginstal Kubernetes pada satu mesin, atau menjalankan bidang kontrol pada mesin dan menambahkan node secara manual, Anda dapat menggunakan alat CLI seperti `kind`, `minikube`, atau `kubeadm` yang terdaftar di bawah Kubernetes Install Tools](#). Untuk menyederhanakan dan mengotomatiskan siklus hidup penuh pembuatan dan pengelolaan kluster, jauh lebih mudah untuk menggunakan alat yang didukung oleh penyedia Kubernetes yang sudah mapan, seperti Amazon EKS atau Amazon EKS Anywhere.

[Di AWS Cloud, Anda dapat membuat kluster Amazon EKS menggunakan alat CLI, seperti `eksctl`, atau alat deklaratif lainnya, seperti Terraform \(lihat Amazon EKS Blueprints for Terraform\)](#). Anda juga dapat membuat cluster dari AWS Management Console. Lihat [fitur Amazon EKS](#) untuk daftar apa yang Anda dapatkan dengan Amazon EKS. Tanggung jawab Kubernetes yang diambil Amazon EKS untuk Anda meliputi:

- Bidang kontrol terkelola —AWS memastikan bahwa kluster Amazon EKS tersedia dan dapat diskalakan karena mengelola bidang kontrol untuk Anda dan membuatnya tersedia di seluruh AWS Availability Zone.
- Manajemen node - Alih-alih menambahkan node secara manual, Anda dapat meminta Amazon EKS membuat node secara otomatis sesuai kebutuhan, menggunakan Grup Node Terkelola (lihat [the section called “Grup simpul terkelola”](#)) atau [Karpenter](#). [Grup Node Terkelola memiliki integrasi dengan Kubernetes Cluster Autoscaling](#). Menggunakan alat manajemen node, Anda dapat memanfaatkan penghematan biaya, dengan hal-hal seperti [Instans Spot](#) dan konsolidasi node, dan ketersediaan, menggunakan fitur [Penjadwalan](#) untuk mengatur bagaimana beban kerja diterapkan dan node dipilih.
- Jaringan cluster — Menggunakan CloudFormation template, `eksctl` menyiapkan jaringan antara control plane dan komponen data plane (node) di cluster Kubernetes. Ini juga mengatur titik akhir di mana komunikasi internal dan eksternal dapat berlangsung. Lihat [De-mystifying jaringan cluster](#)

[untuk node pekerja Amazon EKS](#) untuk detailnya. Komunikasi antar Pod di Amazon EKS dilakukan dengan menggunakan Amazon EKS Pod Identities (lihat [the section called “Identitas Pod”](#)), yang menyediakan sarana untuk membiarkan Pod memanfaatkan metode AWS cloud untuk mengelola kredensial dan izin.

- Add-On — Amazon EKS menyelamatkan Anda dari keharusan membangun dan menambahkan komponen perangkat lunak yang biasa digunakan untuk mendukung kluster Kubernetes. Misalnya, ketika Anda membuat kluster Amazon EKS dari AWS Management Console, secara otomatis akan menambahkan Amazon EKS kube-proxy ([the section called “kube-proxy”](#)), plugin Amazon VPC CNI untuk Kubernetes ([the section called “Amazon VPC CNI”](#)), dan add-on CoreDNS ([the section called “CoreDNS”](#)). Lihat lebih lanjut [the section called “Add-on Amazon EKS”](#) lanjut tentang add-on ini, termasuk daftar yang tersedia.

Untuk menjalankan cluster Anda di komputer dan jaringan lokal Anda sendiri, Amazon menawarkan [Amazon EKS Anywhere](#). Alih-alih AWS Cloud menjadi penyedia, Anda memiliki pilihan untuk menjalankan Amazon EKS Anywhere di [VMWare vSphere](#), [bare metal](#) ([penyedia Tinkerbell](#)), [Snow CloudStack](#), atau platform [Nutanix](#) menggunakan peralatan Anda sendiri.

Amazon EKS Anywhere didasarkan pada perangkat lunak [Amazon EKS Distro](#) yang sama yang digunakan oleh Amazon EKS. [Namun, Amazon EKS Anywhere mengandalkan implementasi yang berbeda dari antarmuka Kubernetes Cluster API \(CAPI\) untuk mengelola siklus hidup penuh mesin di kluster Amazon EKS Anywhere \(seperti CAPV untuk vSphere dan CAPC for\)](#). CloudStack Karena seluruh cluster berjalan pada peralatan Anda, Anda mengambil tanggung jawab tambahan untuk mengelola bidang kontrol dan mencadangkan datanya (lihat etcd nanti di dokumen ini).

Komponen cluster

Komponen cluster Kubernetes dibagi menjadi dua area utama: control plane dan worker node. [Control Plane Components](#) mengelola cluster dan menyediakan akses ke miliknya APIs. Node pekerja (kadang-kadang hanya disebut sebagai Nodes) menyediakan tempat di mana beban kerja yang sebenarnya dijalankan. [Komponen Node](#) terdiri dari layanan yang berjalan pada setiap node untuk berkomunikasi dengan bidang kontrol dan menjalankan kontainer. Kumpulan node pekerja untuk cluster Anda disebut sebagai Data Plane.

Bidang kontrol

Bidang kontrol terdiri dari satu set layanan yang mengelola cluster. Layanan ini semua mungkin berjalan pada satu komputer atau dapat tersebar di beberapa komputer. Secara internal, ini disebut sebagai Control Plane Instances (CPIs). Bagaimana CPIs dijalankan tergantung pada ukuran cluster

dan persyaratan untuk ketersediaan tinggi. Seiring meningkatnya permintaan di klaster, layanan pesawat kontrol dapat menskalakan untuk menyediakan lebih banyak contoh layanan tersebut, dengan permintaan diseimbangkan beban di antara instans.

Tugas yang dilakukan oleh komponen bidang kontrol Kubernetes meliputi:

- Berkomunikasi dengan komponen cluster (API server) — Server API ([kube-apiserver](#)) mengekspos API Kubernetes sehingga permintaan ke cluster dapat dibuat baik dari dalam maupun luar cluster. Dengan kata lain, permintaan untuk menambah atau mengubah objek klaster (Pod, Services, Nodes, dan sebagainya) dapat berasal dari perintah luar, seperti permintaan dari `kubectl` untuk menjalankan Pod. Demikian juga, permintaan dapat dibuat dari server API ke komponen dalam cluster, seperti query ke `kubelet` layanan untuk status Pod.
- Menyimpan data tentang cluster (penyimpanan nilai **etcd** kunci) — `etcd` Layanan ini menyediakan peran penting untuk melacak keadaan cluster saat ini. Jika `etcd` layanan menjadi tidak dapat diakses, Anda tidak akan dapat memperbarui atau menanyakan status klaster, meskipun beban kerja akan terus berjalan untuk sementara waktu. Untuk alasan itu, cluster kritis biasanya memiliki beberapa instance `etcd` layanan yang seimbang beban yang berjalan pada satu waktu dan melakukan pencadangan berkala dari penyimpanan nilai `etcd` kunci jika terjadi kehilangan atau kerusakan data. Perlu diingat bahwa, di Amazon EKS, ini semua ditangani untuk Anda secara otomatis secara default. Amazon EKS Anywhere menyediakan instruksi untuk [pencadangan dan pemulihan etcd](#). Lihat [Model Data etcd](#) untuk mempelajari cara `etcd` mengelola data.
- Schedule Pods to Nodes (Scheduler) — [Permintaan untuk memulai atau menghentikan Pod di Kubernetes diarahkan ke Kubernetes Scheduler \(kube-scheduler\)](#). Karena sebuah cluster dapat memiliki beberapa node yang mampu menjalankan Pod, maka terserah Scheduler untuk memilih node (atau node, dalam kasus replika) Pod mana yang harus dijalankan. Jika tidak ada kapasitas yang cukup untuk menjalankan Pod yang diminta pada node yang ada, permintaan akan gagal, kecuali Anda telah membuat ketentuan lain. Ketentuan tersebut dapat mencakup mengaktifkan layanan seperti Managed Node Groups ([the section called “Grup simpul terkelola”](#)) atau [Karpenter](#) yang dapat secara otomatis memulai node baru untuk menangani beban kerja.
- Menyimpan komponen dalam keadaan yang diinginkan (Controller Manager) — Kubernetes Controller Manager berjalan sebagai proses daemon ([kube-controller-manager](#)) untuk melihat status klaster dan membuat perubahan pada cluster untuk membangun kembali status yang diharapkan. Secara khusus, ada beberapa pengontrol yang mengawasi objek Kubernetes yang berbeda, yang mencakup `statefulset-controller`, `endpoint-controller`, `cronjob-controller`, `node-controller` dan lainnya.

- **Kelola sumber daya cloud (Cloud Controller Manager)** — Interaksi antara Kubernetes dan penyedia cloud yang melakukan permintaan sumber daya pusat data yang mendasarinya ditangani oleh [Cloud Controller Manager](#) ([cloud-controller-manager](#)). Pengontrol yang dikelola oleh Cloud Controller Manager dapat menyertakan pengontrol rute (untuk menyiapkan rute jaringan cloud), pengontrol layanan (untuk menggunakan layanan penyeimbangan beban cloud), dan pengontrol siklus hidup node (untuk menjaga node tetap sinkron dengan Kubernetes sepanjang siklus hidupnya).

Node Pekerja (bidang data)

Untuk kluster Kubernetes simpul tunggal, beban kerja berjalan pada mesin yang sama dengan bidang kontrol. Namun, konfigurasi yang lebih standar adalah memiliki satu atau lebih sistem komputer terpisah ([Node](#)) yang didedikasikan untuk menjalankan beban kerja Kubernetes.

Ketika Anda pertama kali membuat kluster Kubernetes, beberapa alat pembuatan kluster memungkinkan Anda untuk mengonfigurasi sejumlah node tertentu untuk ditambahkan ke cluster (baik dengan mengidentifikasi sistem komputer yang ada atau dengan meminta penyedia membuat yang baru). Sebelum beban kerja ditambahkan ke sistem tersebut, layanan ditambahkan ke setiap node untuk mengimplementasikan fitur-fitur ini:

- **Manage each node (**kubelet**)** — Server API berkomunikasi dengan layanan [kubelet](#) yang berjalan pada setiap node untuk memastikan bahwa node terdaftar dengan benar dan Pod yang diminta oleh Scheduler sedang berjalan. Kubelet dapat membaca manifes Pod dan mengatur volume penyimpanan atau fitur lain yang dibutuhkan oleh Pod pada sistem lokal. Ini juga dapat memeriksa kesehatan wadah yang berjalan secara lokal.
- **Jalankan kontainer pada sebuah node (container runtime)** — [Container Runtime](#) pada setiap node mengelola kontainer yang diminta untuk setiap Pod yang ditetapkan ke node. Itu berarti ia dapat menarik gambar kontainer dari registri yang sesuai, menjalankan penampung, menghentikannya, dan menanggapi kueri tentang wadah. Runtime kontainer default adalah [containerd](#). Pada Kubernetes 1.24, integrasi khusus Docker ([dockershim](#)) yang dapat digunakan sebagai runtime container dihapus dari Kubernetes. Meskipun Anda masih dapat menggunakan Docker untuk menguji dan menjalankan container di sistem lokal Anda, untuk menggunakan Docker dengan Kubernetes Anda sekarang harus [Menginstal Docker Engine](#) pada setiap node untuk menggunakannya dengan Kubernetes.
- **Mengelola jaringan antar kontainer (**kube-proxy**)** — Untuk dapat mendukung komunikasi antar Pod, Kubernetes menggunakan fitur yang disebut sebagai [Layanan](#) untuk menyiapkan jaringan

Pod yang melacak alamat IP dan port yang terkait dengan Pod tersebut. Layanan [kube-proxy](#) berjalan pada setiap node untuk memungkinkan komunikasi antar Pod berlangsung.

Perluas Cluster

Ada beberapa layanan yang dapat Anda tambahkan ke Kubernetes untuk mendukung cluster, tetapi tidak dijalankan di bidang kontrol. Layanan ini sering berjalan langsung pada node di namespace sistem kube atau di namespace sendiri (seperti yang sering dilakukan dengan penyedia layanan pihak ketiga). Contoh umum adalah layanan CoreDNS, yang menyediakan layanan DNS ke cluster. Lihat [Menemukan layanan bawaan untuk informasi tentang cara melihat layanan](#) kluster mana yang berjalan di kube-system di kluster Anda.

Ada berbagai jenis add-on yang dapat Anda pertimbangkan untuk ditambahkan ke cluster Anda. Agar kluster tetap sehat, Anda dapat menambahkan fitur observabilitas (lihat [Monitor cluster](#)) yang memungkinkan Anda melakukan hal-hal seperti pencatatan, audit, dan metrik. Dengan informasi ini, Anda dapat memecahkan masalah yang terjadi, seringkali melalui antarmuka observabilitas yang sama. [Contoh jenis layanan ini termasuk Amazon GuardDuty, CloudWatch \(lihat the section called “Amazon CloudWatch”\), AWS Distro for, plugin Amazon VPC CNI untuk Kubernetes \(lihat the section called “Amazon VPC CNI”\) OpenTelemetry, dan Grafana Kubernetes Monitoring.](#) Untuk penyimpanan (lihat [Penyimpanan data aplikasi](#)), add-on ke Amazon EKS termasuk Amazon Elastic Block Store CSI Driver (lihat [the section called “Amazon EBS”](#)), Amazon Elastic File System CSI Driver (lihat [the section called “Amazon EFS”](#)), dan beberapa add-on penyimpanan pihak ketiga seperti Amazon FSx untuk driver NetApp ONTAP CSI). [the section called “Amazon FSx untuk NetApp ONTAP”](#)

Untuk daftar yang lebih lengkap dari add-on Amazon EKS yang tersedia, lihat [the section called “Add-on Amazon EKS”](#).

Beban kerja

Kubernetes mendefinisikan [Workload](#) sebagai “sebuah aplikasi yang berjalan di Kubernetes.” Aplikasi tersebut dapat terdiri dari serangkaian layanan mikro yang dijalankan sebagai [Container](#) di [Pod](#), atau dapat dijalankan sebagai pekerjaan batch atau jenis aplikasi lainnya. Tugas Kubernetes adalah memastikan bahwa permintaan yang Anda buat untuk objek yang akan disiapkan atau dikerahkan dilakukan. Sebagai seseorang yang menerapkan aplikasi, Anda harus mempelajari tentang bagaimana kontainer dibuat, bagaimana Pod didefinisikan, dan metode apa yang dapat Anda gunakan untuk menerapkannya.

Kontainer

[Elemen paling dasar dari beban kerja aplikasi yang Anda gunakan dan kelola di Kubernetes adalah sebuah Pod.](#) Sebuah Pod mewakili cara memegang komponen aplikasi serta mendefinisikan spesifikasi yang menggambarkan atribut Pod. Bandingkan ini dengan sesuatu seperti paket RPM atau Deb, yang mengemas perangkat lunak bersama untuk sistem Linux, tetapi tidak berjalan sendiri sebagai entitas.

Karena Pod adalah unit deployable terkecil, biasanya memiliki satu kontainer. Namun, beberapa kontainer dapat berada di dalam Pod jika kontainer digabungkan dengan erat. Misalnya, sebuah kontainer server web mungkin dikemas dalam Pod dengan jenis kontainer [sidecar](#) yang dapat menyediakan logging, monitoring, atau layanan lain yang terkait erat dengan kontainer server web. Dalam hal ini, berada di Pod yang sama memastikan bahwa untuk setiap instance Pod yang sedang berjalan, kedua container selalu berjalan pada node yang sama. Demikian juga, semua kontainer dalam sebuah Pod berbagi lingkungan yang sama, dengan kontainer dalam sebuah Pod berjalan seolah-olah mereka berada di host terisolasi yang sama. Efek dari hal ini adalah bahwa kontainer berbagi satu alamat IP yang menyediakan akses ke Pod dan kontainer dapat berkomunikasi satu sama lain seolah-olah mereka berjalan di localhost mereka sendiri.

Spesifikasi Pod ([PodSpec](#)) menentukan status Pod yang diinginkan. Anda dapat menerapkan Pod individual atau beberapa Pod dengan menggunakan sumber daya beban kerja untuk mengelola Template [Pod](#). Sumber daya beban kerja meliputi [Deployments](#) (untuk mengelola beberapa Replika Pod), [StatefulSets](#) (untuk menyebarkan Pod yang perlu unik, seperti Pod database), dan [DaemonSets](#) (di mana sebuah Pod perlu dijalankan secara terus menerus pada setiap node). Lebih lanjut tentang itu nanti.

Sementara Pod adalah unit terkecil yang Anda gunakan, sebuah kontainer adalah unit terkecil yang Anda buat dan kelola.

Kontainer Bangunan

Pod sebenarnya hanya sebuah struktur di sekitar satu atau lebih kontainer, dengan setiap kontainer itu sendiri memegang sistem file, executable, file konfigurasi, pustaka, dan komponen lain untuk benar-benar menjalankan aplikasi. Karena sebuah perusahaan bernama Docker Inc. pertama kali mempopulerkan kontainer, beberapa orang menyebut kontainer sebagai Kontainer Docker. Namun, [Open Container Initiative](#) telah menetapkan runtime kontainer, gambar, dan metode distribusi untuk industri. Ditambah fakta bahwa kontainer dibuat dari banyak fitur Linux yang ada, yang lain sering menyebut kontainer sebagai Kontainer OCI, Kontainer Linux, atau hanya Kontainer.

Saat Anda membangun wadah, Anda biasanya memulai dengan Dockerfile (secara harfiah dinamai itu). Di dalam Dockerfile itu, Anda mengidentifikasi:

- Sebuah image dasar (base container image) adalah wadah yang biasanya dibangun dari versi minimal dari sistem file sistem operasi (seperti [Red Hat Enterprise Linux](#) atau [Ubuntu](#)) atau sistem minimal yang ditingkatkan untuk menyediakan perangkat lunak untuk menjalankan jenis aplikasi tertentu (seperti aplikasi [nodejs](#) atau python).
- Perangkat lunak aplikasi — Anda dapat menambahkan perangkat lunak aplikasi Anda ke wadah Anda dengan cara yang sama seperti Anda menambahkannya ke sistem Linux. Misalnya, di Dockerfile Anda, Anda dapat menjalankan npm dan yarn menginstal aplikasi Java atau yum dan dnf untuk menginstal paket RPM. Dengan kata lain, menggunakan perintah RUN di Dockerfile, Anda dapat menjalankan perintah apa pun yang tersedia di sistem file gambar dasar Anda untuk menginstal perangkat lunak atau mengkonfigurasi perangkat lunak di dalam gambar kontainer yang dihasilkan.
- Instruksi — [Referensi Dockerfile](#) menjelaskan instruksi yang dapat Anda tambahkan ke Dockerfile saat Anda mengonfigurasinya. Ini termasuk instruksi yang digunakan untuk membangun apa yang ada di wadah itu sendiri (ADD atau COPY file dari sistem lokal), mengidentifikasi perintah untuk dijalankan ketika wadah dijalankan (CMD atau ENTRYPOINT), dan menghubungkan wadah ke sistem yang dijalkannya (dengan mengidentifikasi USER untuk dijalankan sebagai, lokal VOLUME untuk dipasang, atau port ke EXPOSE).

Sementara docker perintah dan layanan secara tradisional telah digunakan untuk membangun container (docker build), alat lain yang tersedia untuk membangun gambar kontainer termasuk [podman](#) dan [nerdctl](#). Lihat [Membangun Gambar Kontainer yang Lebih Baik](#) atau [Ikhtisar Docker Build](#) untuk mempelajari tentang membangun kontainer.

Menyimpan Wadah

Setelah Anda membangun gambar kontainer Anda, Anda dapat menyimpannya dalam [registri distribusi kontainer di workstation Anda atau di registri](#) kontainer publik. Menjalankan registri kontainer pribadi di workstation Anda memungkinkan Anda menyimpan gambar kontainer secara lokal, membuatnya tersedia untuk Anda.

Untuk menyimpan gambar kontainer dengan cara yang lebih umum, Anda dapat mendorongnya ke registri kontainer publik. Registries kontainer publik menyediakan lokasi sentral untuk menyimpan dan mendistribusikan gambar kontainer. Contoh pendaftar kontainer publik termasuk [Amazon Elastic Container Registry](#), [registri Red Hat Quay](#), dan registri [Docker](#) Hub.

Saat menjalankan beban kerja kontainer di Amazon Elastic Kubernetes Service (Amazon EKS), kami sarankan untuk menarik salinan Gambar Resmi Docker yang disimpan di Amazon Elastic Container Registry. Amazon ECR telah menyimpan gambar-gambar ini sejak 2021. Anda dapat mencari gambar kontainer populer di [Galeri Publik Amazon ECR](#), dan khusus untuk gambar Docker Hub, Anda dapat mencari Galeri Docker [Amazon ECR](#).

Menjalankan kontainer

Karena kontainer dibangun dalam format standar, kontainer dapat berjalan di mesin apa pun yang dapat menjalankan runtime kontainer (seperti Docker) dan yang isinya cocok dengan arsitektur mesin lokal (seperti x86_64 atau arm). Untuk menguji kontainer atau menjalankannya di desktop lokal Anda, Anda dapat menggunakan `docker run` atau `podman run` perintah untuk memulai penampung di localhost. Namun, untuk Kubernetes, setiap node pekerja memiliki runtime container yang di-deploy dan terserah Kubernetes untuk meminta sebuah node menjalankan sebuah container.

Setelah wadah ditugaskan untuk berjalan pada node, node akan melihat apakah versi yang diminta dari gambar kontainer sudah ada di node. Jika tidak, Kubernetes memberi tahu runtime container untuk menarik container itu dari registri container yang sesuai, lalu jalankan container tersebut secara lokal. Perlu diingat bahwa image kontainer mengacu pada paket perangkat lunak yang dipindahkan antara laptop Anda, registri kontainer, dan node Kubernetes. Sebuah wadah mengacu pada instance yang sedang berjalan dari gambar itu.

Pod

Setelah kontainer Anda siap, bekerja dengan Pod termasuk mengonfigurasi, menerapkan, dan membuat Pod dapat diakses.

Mengkonfigurasi Pod

Ketika Anda mendefinisikan sebuah Pod, Anda menetapkan satu set atribut untuk itu. Atribut-atribut tersebut harus menyertakan setidaknya nama Pod dan image container yang akan dijalankan. Namun, ada banyak hal lain yang ingin Anda konfigurasikan dengan definisi Pod Anda juga (lihat [PodSpec](#) halaman untuk detail tentang apa yang bisa masuk ke Pod). Ini termasuk:

- Penyimpanan — Ketika wadah yang sedang berjalan dihentikan dan dihapus, penyimpanan data dalam wadah itu akan hilang, kecuali jika Anda mengatur penyimpanan yang lebih permanen. [Kubernetes mendukung berbagai jenis penyimpanan dan mengabstraksinya di bawah payung Volume. Jenis penyimpanan termasuk CephFS, NFS, iSCSI, dan lainnya.](#) Anda bahkan dapat menggunakan [perangkat blok lokal](#) dari komputer lokal. Dengan salah satu jenis penyimpanan

yang tersedia dari cluster Anda, Anda dapat memasang volume penyimpanan ke titik pemasangan yang dipilih di sistem file container Anda. [Volume Persisten](#) adalah salah satu yang terus ada setelah Pod dihapus, sementara [Volume Ephemeral](#) dihapus ketika Pod dihapus. Jika administrator kluster Anda membuat [kelas penyimpanan](#) yang berbeda untuk kluster Anda, Anda mungkin memiliki opsi untuk memilih atribut penyimpanan yang Anda gunakan, seperti apakah volume dihapus atau direklamasi setelah digunakan, apakah itu akan diperluas jika lebih banyak ruang diperlukan, dan bahkan apakah itu memenuhi persyaratan kinerja tertentu.

- **Secrets** — Dengan membuat [Secrets](#) tersedia untuk container dalam spesifikasi Pod, Anda dapat memberikan izin yang dibutuhkan container tersebut untuk mengakses sistem file, basis data, atau aset lain yang dilindungi. Kunci, kata sandi, dan token adalah salah satu item yang dapat disimpan sebagai rahasia. Menggunakan rahasia membuatnya sehingga Anda tidak perlu menyimpan informasi ini dalam gambar kontainer, tetapi hanya perlu membuat rahasia tersedia untuk menjalankan wadah. Mirip dengan Rahasia adalah [ConfigMaps](#). A ConfigMap cenderung menyimpan informasi yang kurang penting, seperti pasangan nilai kunci untuk mengonfigurasi layanan.
- **Sumber daya kontainer** — Objek untuk mengkonfigurasi kontainer lebih lanjut dapat berbentuk konfigurasi sumber daya. Untuk setiap kontainer, Anda dapat meminta jumlah memori dan CPU yang dapat digunakan, serta menempatkan batas jumlah total sumber daya yang dapat digunakan wadah. Lihat [Manajemen Sumber Daya untuk Pod dan Container](#) untuk contoh.
- **Gangguan** — Pod dapat terganggu secara tidak sengaja (sebuah node turun) atau secara sukarela (upgrade diinginkan). Dengan mengonfigurasi [anggaran gangguan Pod](#), Anda dapat mengontrol seberapa tersedia aplikasi Anda saat terjadi gangguan. Lihat [Menentukan Anggaran Gangguan](#) untuk aplikasi Anda untuk contoh.
- **Namespaces** — Kubernetes menyediakan berbagai cara untuk mengisolasi komponen dan beban kerja Kubernetes satu sama lain. Menjalankan semua Pod untuk aplikasi tertentu di [Namespace](#) yang sama adalah cara umum untuk mengamankan dan mengelola Pod tersebut bersama-sama. Anda dapat membuat namespace sendiri untuk digunakan atau memilih untuk tidak menunjukkan namespace (yang menyebabkan Kubernetes menggunakan namespace). default [Komponen bidang kontrol Kubernetes biasanya berjalan di namespace sistem kube](#).

Konfigurasi yang baru saja dijelaskan biasanya dikumpulkan bersama dalam file YAMB untuk diterapkan ke cluster Kubernetes. Untuk kluster Kubernetes pribadi, Anda mungkin hanya menyimpan file YAMAL ini di sistem lokal Anda. Namun, dengan kluster dan beban kerja yang lebih kritis, [GitOps](#) adalah cara populer untuk mengotomatiskan penyimpanan dan pembaruan pada beban kerja dan sumber daya infrastruktur Kubernetes.

Objek yang digunakan untuk mengumpulkan dan menyebarkan informasi Pod ditentukan oleh salah satu metode penerapan berikut.

Menerapkan Pod

Metode yang akan Anda pilih untuk menerapkan Pod tergantung pada jenis aplikasi yang Anda rencanakan untuk dijalankan dengan Pod tersebut. Berikut adalah beberapa pilihan Anda:

- **Aplikasi stateless** — Aplikasi stateless tidak menyimpan data sesi klien, jadi sesi lain tidak perlu merujuk kembali ke apa yang terjadi pada sesi sebelumnya. Ini membuatnya lebih mudah untuk mengganti Pod dengan yang baru jika menjadi tidak sehat atau memindahkannya tanpa menyimpan status. [Jika Anda menjalankan aplikasi stateless \(seperti server web\), Anda dapat menggunakan Deployment untuk menyebarkan Pod dan. ReplicaSets](#) A ReplicaSet mendefinisikan berapa banyak instance dari sebuah Pod yang ingin Anda jalankan secara bersamaan. Meskipun Anda dapat menjalankan ReplicaSet secara langsung, adalah umum untuk menjalankan replika secara langsung di dalam Deployment, untuk menentukan berapa banyak replika Pod yang harus dijalankan pada satu waktu.
- **Aplikasi stateful** — Aplikasi stateful adalah aplikasi di mana identitas Pod dan urutan peluncuran Pod adalah penting. Aplikasi ini membutuhkan penyimpanan persisten yang stabil dan perlu digunakan dan diskalakan secara konsisten. Untuk menerapkan aplikasi stateful di Kubernetes, Anda dapat menggunakannya. [StatefulSets](#) Contoh aplikasi yang biasanya StatefulSet dijalankan sebagai database. Dalam a StatefulSet, Anda dapat menentukan replika, Pod dan kontainernya, volume penyimpanan yang akan dipasang, dan lokasi dalam wadah tempat data disimpan. Lihat [Menjalankan Aplikasi Stateful yang Direplikasi](#) untuk contoh database yang digunakan sebagai file. ReplicaSet
- **Aplikasi per-node** — Ada kalanya Anda ingin menjalankan aplikasi pada setiap node di cluster Kubernetes Anda. Misalnya, pusat data Anda mungkin mengharuskan setiap komputer menjalankan aplikasi pemantauan atau layanan akses jarak jauh tertentu. Untuk Kubernetes, Anda dapat menggunakan a [DaemonSet](#) untuk memastikan bahwa aplikasi yang dipilih berjalan pada setiap node di cluster Anda.
- **Aplikasi berjalan hingga selesai** — Ada beberapa aplikasi yang ingin Anda jalankan untuk menyelesaikan tugas tertentu. Ini bisa termasuk yang menjalankan laporan status bulanan atau membersihkan data lama. Objek [Job](#) dapat digunakan untuk menyiapkan aplikasi untuk memulai dan menjalankan, lalu keluar ketika tugas selesai. [CronJob](#) Objek memungkinkan Anda mengatur aplikasi untuk berjalan pada jam, menit, hari tertentu dalam sebulan, bulan, atau hari dalam seminggu, menggunakan struktur yang ditentukan oleh format [crontab](#) Linux.

Membuat aplikasi dapat diakses dari jaringan

Dengan aplikasi yang sering digunakan sebagai satu set layanan mikro yang berpindah ke tempat yang berbeda, Kubernetes membutuhkan cara agar layanan mikro tersebut dapat menemukan satu sama lain. Selain itu, bagi orang lain untuk mengakses aplikasi di luar kluster Kubernetes, Kubernetes membutuhkan cara untuk mengekspos aplikasi itu di alamat dan port luar. Fitur-fitur terkait jaringan ini dilakukan dengan objek Service dan Ingress, masing-masing:

- **Services** — Karena Pod dapat berpindah ke node dan alamat yang berbeda, Pod lain yang perlu berkomunikasi dengan Pod pertama dapat merasa sulit untuk menemukan di mana Pod itu berada. [Untuk mengatasi masalah ini, Kubernetes memungkinkan Anda mewakili aplikasi sebagai Layanan](#). Dengan Service, Anda dapat mengidentifikasi Pod atau kumpulan Pod dengan nama tertentu, kemudian menunjukkan port apa yang mengekspos layanan aplikasi tersebut dari Pod dan port apa yang dapat digunakan aplikasi lain untuk menghubungi layanan tersebut. Pod lain di dalam kluster dapat dengan mudah meminta Service dengan nama dan Kubernetes akan mengarahkan permintaan tersebut ke port yang tepat untuk sebuah instance dari Pod yang menjalankan layanan tersebut.
- **Ingress** — [Ingress](#) adalah apa yang dapat membuat aplikasi yang diwakili oleh Kubernetes Services tersedia untuk klien yang berada di luar kluster. Fitur dasar Ingress termasuk penyeimbang beban (dikelola oleh Ingress), pengontrol Ingress, dan aturan untuk permintaan perutean dari pengontrol ke Layanan. Ada beberapa [Ingress Controller](#) yang dapat Anda pilih dengan Kubernetes.

Langkah selanjutnya

Memahami konsep dasar Kubernetes dan bagaimana kaitannya dengan Amazon EKS akan membantu Anda menavigasi dokumentasi [Amazon EKS dan dokumentasi Kubernetes](#) untuk menemukan informasi yang Anda perlukan untuk mengelola kluster Amazon EKS dan menerapkan beban kerja ke kluster tersebut. Untuk mulai menggunakan Amazon EKS, pilih dari yang berikut ini:

- [the section called “Buat cluster \(eksctl\)”](#)
- [the section called “Membuat kluster”](#)
- [the section called “Contoh penyebaran \(Linux\)”](#)
- [Manajemen kluster](#)

Menerapkan kluster Amazon EKS di seluruh lingkungan cloud dan lokal

Memahami opsi penerapan Amazon EKS

Amazon Elastic Kubernetes Service (Amazon EKS) adalah layanan Kubernetes yang dikelola sepenuhnya yang memungkinkan Anda menjalankan Kubernetes dengan mulus di cloud dan di lingkungan lokal Anda.

Di cloud, Amazon EKS mengotomatiskan manajemen infrastruktur kluster Kubernetes untuk bidang kontrol dan node Kubernetes. Ini penting untuk menjadwalkan kontainer, mengelola ketersediaan aplikasi, menskalakan sumber daya secara dinamis, mengoptimalkan komputasi, menyimpan data cluster, dan melakukan fungsi penting lainnya. Dengan Amazon EKS, Anda mendapatkan kinerja, skalabilitas, keandalan, dan ketersediaan AWS infrastruktur yang kuat, bersama dengan integrasi asli dengan layanan AWS jaringan, keamanan, penyimpanan, dan observabilitas.

Untuk menyederhanakan menjalankan Kubernetes di lingkungan lokal Anda, Anda dapat menggunakan kluster, fitur, dan alat Amazon EKS yang sama ke atau Amazon EKS [Hybrid Nodes pada infrastruktur Anda sendiri, atau Anda dapat menggunakan the section called “Simpul” Amazon EKS](#) Anywhere untuk lingkungan [air-gapped](#) mandiri.

Amazon EKS di cloud

Anda dapat menggunakan Amazon EKS dengan komputasi di AWS Wilayah, AWS Local Zones, dan AWS Wavelength Zones. Dengan Amazon EKS di cloud, keamanan, skalabilitas, dan ketersediaan pesawat kontrol Kubernetes sepenuhnya dikelola oleh AWS in the Region. AWS Saat menjalankan aplikasi dengan komputasi di AWS Wilayah, Anda mendapatkan fitur Amazon EKS yang luas dan lengkap, termasuk AWS Amazon EKS Auto Mode, yang sepenuhnya mengotomatiskan manajemen infrastruktur kluster Kubernetes untuk komputasi, penyimpanan, dan jaringan dengan satu klik. AWS Saat menjalankan aplikasi dengan komputasi di AWS Local Zones dan AWS Wavelength Zones, Anda dapat menggunakan node yang dikelola sendiri Amazon EKS untuk menghubungkan instans Amazon untuk komputasi cluster Anda dan dapat menggunakan layanan EC2 lain yang tersedia di Local Zones dan Wavelength Zones. AWS AWS AWS Untuk informasi selengkapnya, lihat fitur [AWS Local Zones](#) dan fitur [AWS Wavelength](#) Zones.

	Amazon EKS di AWS Wilayah	Amazon EKS di Zona Lokal/ Panjang Gelombang
Manajemen pesawat kontrol Kubernetes	AWS-dikelola	AWS-dikelola
Kubernetes mengontrol lokasi pesawat	AWS Daerah	AWS Daerah
Pesawat data Kubernetes	• Mode Otomatis Amazon EKS • Grup Node Terkelola Amazon EKS • Node yang EC2 dikelola sendiri Amazon • AWS Fargate	• Grup Node Terkelola Amazon EKS (Hanya Local Zones) • Node yang EC2 dikelola sendiri Amazon
Lokasi pesawat data Kubernetes	AWS Daerah	AWS Zona Lokal atau Wavelength

Amazon EKS di pusat data atau lingkungan edge

Jika Anda perlu menjalankan aplikasi di pusat data atau lingkungan edge Anda sendiri, Anda dapat menggunakan [Amazon EKS di AWS Outposts](#) atau [Amazon EKS Hybrid Nodes](#). Anda dapat menggunakan node yang dikelola sendiri dengan EC2 instans Amazon di AWS Outposts untuk komputasi klaster, atau Anda dapat menggunakan Amazon EKS Hybrid Nodes dengan infrastruktur lokal atau edge Anda sendiri untuk komputasi klaster Anda. AWS Outposts adalah infrastruktur AWS terkelola yang Anda jalankan di pusat data atau fasilitas lokasi bersama, sedangkan Amazon EKS Hybrid Nodes berjalan pada mesin fisik atau virtual yang Anda kelola di lingkungan lokal atau edge Anda. Amazon EKS di AWS Outposts dan Amazon EKS Hybrid Nodes memerlukan koneksi yang andal dari lingkungan lokal Anda ke AWS Wilayah, dan Anda dapat menggunakan kluster, fitur, dan alat Amazon EKS yang sama yang Anda gunakan untuk menjalankan aplikasi di cloud. Saat berjalan di AWS Outposts, Anda dapat menerapkan seluruh klaster Kubernetes di Outposts dengan kluster lokal Amazon EKS di AWS Outposts. AWS

	Node Hibrida Amazon EKS	Amazon EKS di AWS Outposts
Manajemen pesawat kontrol Kubernetes	AWS-dikelola	AWS-dikelola
Kubernetes mengontrol lokasi pesawat	AWS Daerah	AWS Daerah atau AWS Outposts
Pesawat data Kubernetes	Mesin fisik atau virtual yang dikelola pelanggan	Node yang EC2 dikelola sendiri Amazon
Lokasi pesawat data Kubernetes	Pusat data pelanggan atau lingkungan tepi	Pusat data pelanggan atau lingkungan tepi

Amazon EKS Anywhere untuk lingkungan yang memiliki jurang udara

[Amazon EKS Anywhere](#) menyederhanakan pengelolaan kluster Kubernetes melalui otomatisasi beban berat yang tidak terdiferensiasi seperti penyiapan infrastruktur dan operasi siklus hidup kluster Kubernetes di lingkungan lokal dan edge. Tidak seperti Amazon EKS, Amazon EKS Anywhere adalah produk yang dikelola pelanggan dan pelanggan bertanggung jawab atas operasi siklus hidup cluster dan pemeliharaan cluster Amazon EKS Anywhere. Amazon EKS Anywhere dibangun di atas sub-proyek Kubernetes Cluster API (CAPI) dan mendukung berbagai infrastruktur termasuk VMware vSphere, bare metal, Nutanix, Apache, dan Snow. CloudStack AWS Amazon EKS Anywhere dapat dijalankan di lingkungan yang memiliki jurang udara dan menawarkan integrasi opsional dengan AWS layanan regional untuk observabilitas dan manajemen identitas. [Untuk menerima dukungan Amazon EKS Anywhere dan akses ke add-on AWS Kubernetes yang dijual, Anda dapat membeli Langganan Amazon EKS Anywhere Enterprise.](#)

	Amazon EKS Anywhere
Manajemen pesawat kontrol Kubernetes	Dikelola pelanggan
Kubernetes mengontrol lokasi pesawat	Pusat data pelanggan atau lingkungan tepi
Pesawat data Kubernetes	Mesin fisik atau virtual yang dikelola pelanggan

	Amazon EKS Anywhere
Lokasi pesawat data Kubernetes	Pusat data pelanggan atau lingkungan tepi

Perkakas Amazon EKS

Anda dapat menggunakan [Amazon EKS Connector](#) untuk mendaftarkan dan menghubungkan kluster Kubernetes yang sesuai dan AWS melihatnya di konsol Amazon EKS. Setelah cluster terhubung, Anda dapat melihat status, konfigurasi, dan beban kerja untuk kluster tersebut di konsol Amazon EKS. Anda dapat menggunakan fitur ini untuk melihat kluster yang terhubung di konsol Amazon EKS, tetapi Konektor Amazon EKS tidak mengaktifkan operasi manajemen atau mutasi untuk kluster yang terhubung melalui konsol Amazon EKS.

[Amazon EKS Distro](#) adalah AWS distribusi komponen Kubernetes yang mendasari yang mendukung semua penawaran Amazon EKS. Ini mencakup komponen inti yang diperlukan untuk kluster Kubernetes yang berfungsi seperti komponen bidang kontrol Kubernetes (etcd, kube-apiserver, kube-scheduler,) dan komponen jaringan (CoreDNS, kube-proxy, plugin CNI). kube-controller-manager Amazon EKS Distro dapat digunakan untuk mengelola sendiri cluster Kubernetes dengan alat pilihan Anda. Penerapan Amazon EKS Distro tidak tercakup oleh Support AWS Plans.

Siapkan untuk menggunakan Amazon EKS

Untuk mempersiapkan manajemen baris perintah kluster Amazon EKS Anda, Anda perlu menginstal beberapa alat. Gunakan yang berikut ini untuk menyiapkan kredensial, membuat dan memodifikasi kluster, dan bekerja dengan cluster setelah dijalankan:

- [Siapkan AWS CLI](#) — Dapatkan AWS CLI untuk mengatur dan mengelola layanan yang Anda butuhkan untuk bekerja dengan kluster Amazon EKS. Secara khusus, Anda memerlukan AWS CLI untuk mengonfigurasi kredensial, tetapi Anda juga membutuhkannya dengan layanan lain. AWS
- [Siapkan kubectl dan eksctl](#) — `eksctl` CLI berinteraksi dengan AWS untuk membuat, memodifikasi, dan menghapus kluster Amazon EKS. Setelah cluster selesai, gunakan `kubectl` perintah open source untuk mengelola objek Kubernetes di dalam kluster Amazon EKS Anda.
- Siapkan lingkungan pengembangan (opsional) - Pertimbangkan untuk menambahkan alat berikut:
 - Alat penyebaran lokal — [Jika Anda baru mengenal Kubernetes, pertimbangkan untuk menginstal alat penyebaran lokal seperti minikube atau kind](#). Alat-alat ini memungkinkan Anda memiliki kluster Amazon EKS di mesin lokal Anda untuk menguji aplikasi.
 - Package manager — [helm](#) adalah manajer paket populer untuk Kubernetes yang menyederhanakan instalasi dan pengelolaan paket yang kompleks. Dengan [Helm](#), lebih mudah untuk menginstal dan mengelola paket seperti AWS Load Balancer Controller di cluster Amazon EKS Anda.

Langkah selanjutnya

- [Mengatur AWS CLI](#)
- [Siapkan kubectl dan eksctl](#)
- [Mulai cepat: Menyebarkan aplikasi web dan menyimpan data](#)

Mengatur AWS CLI

[AWS CLI](#) adalah alat baris perintah untuk bekerja dengan AWS layanan, termasuk Amazon EKS. Ini juga digunakan untuk mengautentikasi pengguna IAM atau peran untuk akses ke kluster Amazon EKS dan AWS sumber daya lainnya dari mesin lokal Anda. Untuk menyediakan sumber daya AWS dari baris perintah, Anda perlu mendapatkan ID kunci AWS akses dan kunci rahasia untuk digunakan di baris perintah. Maka Anda perlu mengkonfigurasi kredensial ini di CLI AWS . Jika Anda

belum menginstal AWS CLI, lihat [Menginstal atau memperbarui versi terbaru AWS CLI di Panduan Pengguna Antarmuka Baris AWS Perintah](#).

Untuk membuat kunci akses

1. Masuk ke [AWS Management Console](#).
2. Untuk akun pengguna tunggal atau beberapa pengguna:
 - Akun pengguna tunggal —:: Di kanan atas, pilih nama AWS pengguna Anda untuk membuka menu navigasi. Misalnya, pilih **webadmin**.
 - Akun multi-pengguna —:: Pilih IAM dari daftar layanan. Dari Dasbor IAM, pilih Pengguna, dan pilih nama pengguna.
3. Pilih Kredensial keamanan.
4. Di bawah tombol Access, pilih Create Access Key.
5. Pilih Command Line Interface (CLI), lalu pilih Berikutnya.
6. Pilih Buat access key.
7. Pilih Unduh file.csv.

Untuk mengkonfigurasi AWS CLI

Setelah menginstal AWS CLI, lakukan langkah-langkah berikut untuk mengkonfigurasinya. Untuk informasi selengkapnya, lihat [Mengkonfigurasi AWS CLI](#) di Panduan Pengguna Antarmuka Baris AWS Perintah.

1. Di jendela terminal, masukkan perintah berikut:

```
aws configure
```

Secara opsional, Anda dapat mengonfigurasi profil bernama, seperti `--profile cluster-admin`. Jika Anda mengonfigurasi profil bernama di AWS CLI, Anda harus selalu meneruskan bendera ini di perintah berikutnya.

2. Masukkan AWS kredensial Anda. Sebagai contoh:

```
Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Default region name [None]: region-code
```

```
Default output format [None]: json
```

Untuk mendapatkan token keamanan

Jika diperlukan, jalankan perintah berikut untuk mendapatkan token keamanan baru untuk AWS CLI. Untuk informasi selengkapnya, lihat [get-session-token](#) di Referensi Perintah AWS CLI.

Secara default, token berlaku selama 15 menit. Untuk mengubah batas waktu sesi default, berikan `--duration-seconds` tanda. Sebagai contoh:

```
aws sts get-session-token --duration-seconds 3600
```

Perintah ini mengembalikan kredensi keamanan sementara untuk sesi CLI AWS . Anda akan melihat output respons berikut:

```
{
  "Credentials": {
    "AccessKeyId": "ASIA5FTRU3LOEXAMPLE",
    "SecretAccessKey": "JnKgvmfQUD9mNsPoi9IbxAYEXAMPLE",
    "SessionToken": "VERYLONGSESSIONTOKENSTRING",
    "Expiration": "2023-02-17T03:14:24+00:00"
  }
}
```

Untuk memverifikasi identitas pengguna

Jika diperlukan, jalankan perintah berikut untuk memverifikasi AWS kredensial identitas pengguna IAM Anda (seperti *ClusterAdmin*) untuk sesi terminal.

```
aws sts get-caller-identity
```

Perintah ini mengembalikan Amazon Resource Name (ARN) dari entitas IAM yang dikonfigurasi untuk CLI. AWS Anda akan melihat contoh output respon berikut:

```
{
  "UserId": "AKIAIOSFODNN7EXAMPLE",
  "Account": "01234567890",
  "Arn": "arn:aws:iam::01234567890:user/ClusterAdmin"
```

```
}
```

Langkah selanjutnya

- [Siapkan kubectl dan eksctl](#)
- [Quickstart: Menyebarkan aplikasi web dan menyimpan data](#)

Mengatur **kubectl** dan **eksctl**

Setelah AWS CLI diinstal, ada dua alat lain yang harus Anda instal untuk membuat dan mengelola cluster Kubernetes Anda:

- **kubectl**: Alat baris **kubectl** perintah adalah alat utama yang akan Anda gunakan untuk mengelola sumber daya di dalam klaster Kubernetes Anda. Halaman ini menjelaskan cara mengunduh dan menyiapkan **kubectl** biner yang cocok dengan versi klaster Kubernetes Anda. Lihat [Menginstal atau memperbarui kubectl](#).
- **eksctl**: Alat baris **eksctl** perintah dibuat untuk membuat kluster EKS di AWS cloud atau lokal (dengan EKS Anywhere), serta memodifikasi dan menghapus cluster tersebut. Lihat [Instal eksctl](#).

Instal atau perbarui **kubectl**

Topik ini membantu Anda mengunduh dan menginstal, atau memperbarui, **kubectl** biner di perangkat Anda. Biner identik dengan [versi komunitas hulu](#). Biner ini tidak unik untuk Amazon EKS atau AWS. Gunakan langkah-langkah di bawah ini untuk mendapatkan versi spesifik **kubectl** yang Anda butuhkan, meskipun banyak pembangun hanya menjalankan `brew install kubectl` untuk menginstalnya.

Note

Anda harus menggunakan versi **kubectl** yang tidak lebih dari satu perbedaan kecil dari bidang kendali Amazon EKS klaster Anda. Misalnya, 1.32 **kubectl** klien bekerja dengan Kubernetes 1.31, 1.32, dan cluster. 1.33

Langkah 1: Periksa apakah **kubectl** sudah diinstal

Tentukan apakah Anda sudah **kubectl** menginstal pada perangkat Anda.

```
kubectl version --client
```

Jika Anda telah `kubectl` menginstal di jalur perangkat Anda, contoh output mencakup informasi yang mirip dengan berikut ini. Jika Anda ingin memperbarui versi yang saat ini telah Anda instal dengan versi yang lebih baru, selesaikan langkah berikutnya, pastikan untuk menginstal versi baru di lokasi yang sama dengan versi Anda saat ini.

```
Client Version: v1.31.X-eks-1234567
```

Jika Anda tidak menerima output, maka Anda belum `kubectl` menginstal, atau tidak diinstal di lokasi yang ada di jalur perangkat Anda.

Langkah 2: Instal atau perbarui **kubectl**

Instal atau perbarui `kubectl` pada salah satu sistem operasi berikut:

- [the section called “macOS”](#)
- [the section called “Linux \(amd64\)”](#)
- [the section called “Linux \(arm64\)”](#)
- [the section called “Windows”](#)

Note

Jika unduhan lambat ke AWS Wilayah Anda dari AWS Wilayah yang digunakan di bagian ini, pertimbangkan CloudFront untuk menyiapkan konten di depan. Untuk informasi lebih lanjut, lihat [Memulai CloudFront distribusi dasar](#).

macOS

Ikuti langkah-langkah di bawah ini untuk menginstal `kubectl` di macOS. Langkah-langkahnya meliputi:

- Memilih dan mengunduh biner untuk versi Kubernetes yang Anda inginkan.
- Secara opsional memeriksa checksum biner.
- Menambahkan eksekusi ke izin biner.

- Menyalin biner ke folder di PATH Anda.
- Opsional menambahkan direktori biner ke PATH Anda.

Prosedur:

1. Unduh biner untuk versi Kubernetes cluster Anda dari Amazon S3.

- Kubernetes 1.33

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.0/2025-05-01/bin/darwin/amd64/kubectl
```

- Kubernetes 1.32

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.3/2025-04-17/bin/darwin/amd64/kubectl
```

- Kubernetes 1.31

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.7/2025-04-17/bin/darwin/amd64/kubectl
```

- Kubernetes 1.30

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.11/2025-04-17/bin/darwin/amd64/kubectl
```

- Kubernetes 1.29

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-04-17/bin/darwin/amd64/kubectl
```

- Kubernetes 1.28

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-04-17/bin/darwin/amd64/kubectl
```

- Kubernetes 1.27

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2025-01-10/bin/darwin/amd64/kubectl
```

- Kubernetes 1.26

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/bin/darwin/amd64/kubectl
```

2. (Opsional) Verifikasi biner yang diunduh dengan SHA-256 checksum untuk biner Anda.

a. Unduh SHA-256 checksum untuk versi Kubernetes kluster Anda.

- Kubernetes 1.33

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.0/2025-05-01/bin/darwin/amd64/kubectl.sha256
```

- Kubernetes 1.32

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.3/2025-04-17/bin/darwin/amd64/kubectl.sha256
```

- Kubernetes 1.31

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.7/2025-04-17/bin/darwin/amd64/kubectl.sha256
```

- Kubernetes 1.30

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.11/2025-04-17/bin/darwin/amd64/kubectl.sha256
```

- Kubernetes 1.29

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-04-17/bin/darwin/amd64/kubectl.sha256
```

- Kubernetes 1.28

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-04-17/bin/darwin/amd64/kubectl.sha256
```

- Kubernetes 1.27

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2025-01-10/bin/darwin/amd64/kubectl.sha256
```

- Kubernetes 1.26

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/bin/darwin/amd64/kubectl.sha256
```

- b. Periksa SHA-256 checksum untuk biner yang Anda unduh.

```
openssl sha1 -sha256 kubectl
```

- c. Pastikan checksum yang dihasilkan dalam output cocok dengan checksum di file yang diunduh `kubectl.sha256`.

3. Menerapkan izin eksekusi ke biner.

```
chmod +x ./kubectl
```

4. Salin biner ke folder PATH Anda. Jika Anda telah menginstal versi `kubectl`, maka kami merekomendasikan untuk membuat `$HOME/bin/kubectl` dan pastikan `$HOME/bin` terinstal lebih dulu di `$PATH` Anda.

```
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
```

5. (Opsional) Tambahkan path `$HOME/bin` ke file inisialisasi shell Anda sehingga terkonfigurasi ketika Anda membuka shell.

```
echo 'export PATH=$HOME/bin:$PATH' >> ~/.bash_profile
```

Linux (amd64)

Ikuti langkah-langkah di bawah ini untuk menginstal `kubectl` di Linux (amd64). Langkah-langkahnya meliputi:

- Memilih dan mengunduh biner untuk versi Kubernetes yang Anda inginkan.
- Secara opsional memeriksa checksum biner.
- Menambahkan eksekusi ke izin biner.

- Menyalin biner ke folder di PATH Anda.
- Opsional menambahkan direktori biner ke PATH Anda.

Prosedur:

1. Unduh `kubectl` biner untuk versi Kubernetes cluster Anda dari Amazon S3.

- Kubernetes 1.33

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.0/2025-05-01/bin/linux/amd64/kubectl
```

- Kubernetes 1.32

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.3/2025-04-17/bin/linux/amd64/kubectl
```

- Kubernetes 1.31

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.7/2025-04-17/bin/linux/amd64/kubectl
```

- Kubernetes 1.30

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.11/2025-04-17/bin/linux/amd64/kubectl
```

- Kubernetes 1.29

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-04-17/bin/linux/amd64/kubectl
```

- Kubernetes 1.28

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-04-17/bin/linux/amd64/kubectl
```

- Kubernetes 1.27

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2024-12-12/bin/linux/amd64/kubectl
```

- Kubernetes 1.26

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/bin/linux/amd64/kubectl
```

2. (Opsional) Verifikasi biner yang diunduh dengan SHA-256 checksum untuk biner Anda.

- a. Unduh SHA-256 checksum untuk versi Kubernetes cluster Anda dari Amazon S3 menggunakan perintah untuk platform perangkat keras perangkat Anda.

- Kubernetes 1.33

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.0/2025-05-01/bin/linux/amd64/kubectl.sha256
```

- Kubernetes 1.32

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.3/2025-04-17/bin/linux/amd64/kubectl.sha256
```

- Kubernetes 1.31

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.7/2025-04-17/bin/linux/amd64/kubectl.sha256
```

- Kubernetes 1.30

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.11/2025-04-17/bin/linux/amd64/kubectl.sha256
```

- Kubernetes 1.29

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-04-17/bin/linux/amd64/kubectl.sha256
```

- Kubernetes 1.28

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-04-17/bin/linux/amd64/kubectl.sha256
```

- Kubernetes 1.27

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2024-12-12/bin/linux/amd64/kubectl.sha256
```

- Kubernetes 1.26

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/bin/linux/amd64/kubectl.sha256
```

b. Periksa SHA-256 checksum untuk biner yang Anda unduh dengan salah satu perintah berikut.

```
sha256sum -c kubectl.sha256
```

atau

```
openssl sha1 -sha256 kubectl
```

c. Untuk yang pertama, Anda harus melihat `kubectl: OK`, untuk yang kedua, Anda dapat memeriksa bahwa checksum yang dihasilkan dalam output cocok dengan checksum di file yang diunduh `kubectl.sha256`.

3. Menerapkan izin eksekusi ke biner.

```
chmod +x ./kubectl
```

4. Salin biner ke folder PATH Anda. Jika Anda telah menginstal versi `kubectl`, maka kami merekomendasikan untuk membuat `$HOME/bin/kubectl` dan pastikan `$HOME/bin` terinstal lebih dulu di `$PATH` Anda.

```
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
```

5. (Opsional) Tambahkan path `$HOME/bin` ke file inisialisasi shell Anda sehingga terkonfigurasi ketika Anda membuka shell.

 Note

Langkah ini mengasumsikan Anda menggunakan shell Bash; jika Anda menggunakan shell lain, ubah perintah untuk menggunakan file inisialisasi shell spesifik Anda.

```
echo 'export PATH=$HOME/bin:$PATH' >> ~/.bashrc
```

Linux (arm64)

Ikuti langkah-langkah di bawah ini untuk menginstal `kubectl` di Linux (arm64). Langkah-langkahnya meliputi:

- Memilih dan mengunduh biner untuk versi Kubernetes yang Anda inginkan.
- Secara opsional memeriksa checksum biner.
- Menambahkan eksekusi ke izin biner.
- Menyalin biner ke folder di PATH Anda.
- Opsional menambahkan direktori biner ke PATH Anda.

Prosedur:

1. Unduh `kubectl` biner untuk versi Kubernetes cluster Anda dari Amazon S3.

- Kubernetes 1.33

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.0/2025-05-01/bin/linux/arm64/kubectl
```

- Kubernetes 1.32

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.3/2025-04-17/bin/linux/arm64/kubectl
```

- Kubernetes 1.31

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.7/2025-04-17/bin/linux/arm64/kubectl
```

- Kubernetes 1.30

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.11/2025-04-17/bin/linux/arm64/kubectl
```

- Kubernetes 1.29

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-04-17/bin/linux/arm64/kubectl
```

- Kubernetes 1.28

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-04-17/bin/linux/arm64/kubectl
```

- Kubernetes 1.27

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2024-12-12/bin/linux/arm64/kubectl
```

- Kubernetes 1.26

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/bin/linux/arm64/kubectl
```

2. (Optional) Verifikasi biner yang diunduh dengan SHA-256 checksum untuk biner Anda.

- a. Unduh SHA-256 checksum untuk versi Kubernetes cluster Anda dari Amazon S3 menggunakan perintah untuk platform perangkat keras perangkat Anda.

- Kubernetes 1.33

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.0/2025-05-01/bin/linux/arm64/kubectl.sha256
```

- Kubernetes 1.32

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.3/2025-04-17/bin/linux/arm64/kubectl.sha256
```

- Kubernetes 1.31

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.7/2025-04-17/bin/linux/arm64/kubectl.sha256
```

- Kubernetes 1.30

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.11/2025-04-17/bin/
linux/arm64/kubectl.sha256
```

- Kubernetes 1.29

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-04-17/bin/
linux/arm64/kubectl.sha256
```

- Kubernetes 1.28

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-04-17/bin/
linux/arm64/kubectl.sha256
```

- Kubernetes 1.27

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2024-12-12/bin/
linux/arm64/kubectl.sha256
```

- Kubernetes 1.26

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/bin/
linux/arm64/kubectl.sha256
```

b. Periksa SHA-256 checksum untuk biner yang Anda unduh dengan salah satu perintah berikut.

```
sha256sum -c kubectl.sha256
```

atau

```
openssl sha1 -sha256 kubectl
```

c. Untuk yang pertama, Anda harus melihat `kubectl`: OK, untuk yang kedua, Anda dapat memeriksa bahwa checksum yang dihasilkan dalam output cocok dengan checksum di file yang diunduh `kubectl.sha256`.

3. Menerapkan izin eksekusi ke biner.

```
chmod +x ./kubectl
```

4. Salin biner ke folder PATH Anda. Jika Anda telah menginstal versi `kubectl`, maka kami merekomendasikan untuk membuat `$HOME/bin/kubectl` dan pastikan `$HOME/bin` terinstal lebih dulu di `$PATH` Anda.

```
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
```

5. (Opsional) Tambahkan path `$HOME/bin` ke file inisialisasi shell Anda sehingga terkonfigurasi ketika Anda membuka shell.

Note

Langkah ini mengasumsikan Anda menggunakan shell Bash; jika Anda menggunakan shell lain, ubah perintah untuk menggunakan file inisialisasi shell spesifik Anda.

```
echo 'export PATH=$HOME/bin:$PATH' >> ~/.bashrc
```

Windows

Ikuti langkah-langkah di bawah ini untuk menginstal `kubectl` di Windows. Langkah-langkahnya meliputi:

- Memilih dan mengunduh biner untuk versi Kubernetes yang Anda inginkan.
- Secara opsional memeriksa checksum biner.
- Menyalin biner ke folder di PATH Anda.
- Opsional menambahkan direktori biner ke PATH Anda.

Prosedur:

1. Buka PowerShell terminal.
2. Unduh `kubectl` biner untuk versi Kubernetes cluster Anda dari Amazon S3.
 - Kubernetes 1.33

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.0/2025-05-01/bin/windows/amd64/kubectl.exe
```

- Kubernetes 1.32

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.3/2025-04-17/bin/windows/amd64/kubectl.exe
```

- Kubernetes 1.31

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.7/2025-04-17/bin/windows/amd64/kubectl.exe
```

- Kubernetes 1.30

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.11/2025-04-17/bin/windows/amd64/kubectl.exe
```

- Kubernetes 1.29

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-04-17/bin/windows/amd64/kubectl.exe
```

- Kubernetes 1.28

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-04-17/bin/windows/amd64/kubectl.exe
```

- Kubernetes 1.27

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2024-12-12/bin/windows/amd64/kubectl.exe
```

- Kubernetes 1.26

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/bin/windows/amd64/kubectl.exe
```

3. (Opsional) Verifikasi biner yang diunduh dengan SHA-256 checksum untuk biner Anda.

a. Unduh SHA-256 checksum untuk versi Kubernetes cluster Anda untuk Windows.

- Kubernetes 1.33


```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.0/2025-05-01/bin/windows/amd64/kubectl.exe.sha256
```

- **Kubernetes 1.32**

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.3/2025-04-17/bin/windows/amd64/kubectl.exe.sha256
```

- **Kubernetes 1.31**

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.7/2025-04-17/bin/windows/amd64/kubectl.exe.sha256
```

- **Kubernetes 1.30**

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.11/2025-04-17/bin/windows/amd64/kubectl.exe.sha256
```

- **Kubernetes 1.29**

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-04-17/bin/windows/amd64/kubectl.exe.sha256
```

- **Kubernetes 1.28**

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-04-17/bin/windows/amd64/kubectl.exe.sha256
```

- **Kubernetes 1.27**

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2024-12-12/bin/windows/amd64/kubectl.exe.sha256
```

- **Kubernetes 1.26**

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/bin/windows/amd64/kubectl.exe.sha256
```

b. Periksa SHA-256 checksum untuk biner yang Anda unduh.

```
Get-FileHash kubect1.exe
```

- c. Pastikan checksum yang dihasilkan dalam output cocok dengan checksum di file yang diunduh `kubect1.sha256`. PowerShell Outputnya harus berupa string karakter yang setara huruf besar.
4. Salin biner ke folder PATH Anda. Jika Anda memiliki direktori yang ada di dalam Anda PATH yang Anda gunakan untuk utilitas baris perintah, salin biner ke direktori itu. Jika tidak, selesaikan langkah-langkah berikut.
 - a. Membuat direktori baru untuk binari baris perintah Anda, seperti `C:\bin`.
 - b. Salin biner `kubect1.exe` ke direktori baru Anda.
 - c. Edit variabel PATH lingkungan pengguna atau sistem Anda untuk menambahkan direktori baru ke direktori Anda PATH.
 - d. Tutup PowerShell terminal Anda dan buka yang baru untuk mengambil PATH variabel baru.
5. Setelah Anda menginstall `kubect1`, Anda dapat memverifikasi versinya.

```
kubect1 version --client
```

6. Saat pertama kali menginstall `kubect1`, itu belum dikonfigurasi untuk berkomunikasi dengan server apa pun. Kami akan membahas konfigurasi ini sesuai kebutuhan dalam prosedur lain. Jika Anda perlu memperbarui konfigurasi untuk berkomunikasi dengan cluster tertentu, Anda dapat menjalankan perintah berikut. Ganti *region-code* dengan AWS Wilayah tempat klaster Anda berada. Ganti *my-cluster* dengan nama klaster Anda.

```
aws eks update-kubeconfig --region region-code --name my-cluster
```

7. Pertimbangkan untuk mengonfigurasi penyelesaian otomatis, yang memungkinkan Anda menggunakan tombol tab untuk menyelesaikan `kubect1` subperintah setelah mengetik beberapa huruf pertama. Lihat [pelengkapan otomatis Kubect1](#) di dokumentasi Kubernetes untuk detailnya.

Instal eksctl

`eksctl` CLI digunakan untuk bekerja dengan cluster EKS. Ini mengotomatiskan banyak tugas individu. Lihat [Instalasi](#) dalam `eksctl` dokumentasi untuk petunjuk tentang menginstall `eksctl`. Untuk Linux, gunakan instruksi UNIX.

Saat menggunakan eksctl prinsip keamanan IAM yang Anda gunakan harus memiliki izin untuk bekerja dengan peran Amazon EKS IAM, peran terkait layanan, VPC, AWS CloudFormation dan sumber daya terkait. Untuk informasi selengkapnya, lihat [Tindakan](#) dan [Menggunakan peran terkait layanan](#) di Panduan Pengguna IAM. Anda harus menyelesaikan semua langkah dalam panduan ini sebagai pengguna yang sama. Untuk memeriksa pengguna saat ini, jalankan perintah berikut:

```
aws sts get-caller-identity
```

Langkah selanjutnya

- [Quickstart: Menyebarkan aplikasi web dan menyimpan data](#)

Mulai cepat: Menyebarakan aplikasi web dan menyimpan data

Terapkan aplikasi game dan pertahankan datanya di Amazon EKS

[Tutorial quickstart ini menunjukkan langkah-langkah untuk menyebarakan aplikasi contoh game 2048 dan mempertahankan datanya di cluster Amazon EKS Auto Mode menggunakan eksctl.](#)

Amazon EKS Auto Mode mengotomatiskan tugas rutin untuk penyimpanan blok cluster, jaringan, penyeimbangan beban, dan penskalaan otomatis komputasi.

Saat kami maju, kami akan memandu Anda melalui proses pengaturan cluster. Amazon EKS Auto Mode akan mengotomatiskan tugas untuk membuat node menggunakan instance EC2 terkelola, membuat penyeimbang beban aplikasi, dan membuat volume EBS.

Secara keseluruhan, Anda akan menerapkan beban kerja sampel dengan anotasi khusus yang diperlukan untuk sepenuhnya berintegrasi dengan layanan. AWS

Dalam tutorial ini

Menggunakan template `eksctl` cluster berikut, Anda akan membangun sebuah cluster dengan EKS Auto Mode untuk penyediaan node otomatis.

Konfigurasi VPC Saat menggunakan template cluster `eksctl` yang mengikuti, `eksctl` secara otomatis membuat Virtual Private Cloud (IPv4 VPC) untuk cluster. Secara default, `eksctl` mengonfigurasi VPC yang menangani semua persyaratan jaringan, selain membuat titik akhir publik dan pribadi.

Instance Management EKS Auto Mode secara dinamis menambahkan atau menghapus node di kluster EKS Anda berdasarkan permintaan aplikasi Kubernetes Anda.

Persistensi Data Gunakan kemampuan penyimpanan blok Mode Otomatis EKS untuk memastikan persistensi data aplikasi, bahkan dalam skenario yang melibatkan restart atau kegagalan pod.

Akses Aplikasi Eksternal Gunakan kemampuan load balancing dari Mode Otomatis EKS untuk menyediakan Application Load Balancer (ALB) secara dinamis.

Prasyarat

Sebelum memulai, pastikan Anda memiliki prasyarat berikut yang disiapkan untuk menggunakan Amazon EKS:

- Siapkan AWS CLI dan konfigurasi kredensial
- Instal eksctl
- Instal kubectl

Untuk informasi selengkapnya, lihat [Penyiapan](#).

Konfigurasi cluster

Di bagian ini, Anda akan membuat cluster menggunakan Mode Otomatis EKS untuk penyediaan node dinamis.

Buat `cluster-config.yaml` file dan tempel konten berikut ke dalamnya. Ganti `region-code` dengan Region yang valid, seperti `us-east-1`:

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: web-quickstart
  region: <region-code>

autoModeConfig:
  enabled: true
```

Sekarang, kita siap untuk membuat cluster.

Buat cluster Amazon EKS:

```
eksctl create cluster -f cluster-config.yaml
```

Important

Jika Anda tidak menggunakan eksctl untuk membuat cluster, Anda perlu menandai subnet VPC secara manual.

Buat IngressClass

Buat Kubernetes IngressClass untuk Mode Otomatis EKS. Ini IngressClass mendefinisikan bagaimana Mode Otomatis EKS menangani sumber daya Ingress. Langkah ini mengonfigurasi kemampuan load balancing dari Mode Otomatis EKS. Ketika Anda membuat sumber daya Ingress untuk aplikasi Anda, Mode Otomatis EKS menggunakan ini IngressClass untuk secara otomatis menyediakan dan mengelola load balancer, mengintegrasikan aplikasi Kubernetes Anda dengan layanan load balancing. AWS

Simpan file yaml berikut sebagai `ingressclass.yaml`:

```
apiVersion: networking.k8s.io/v1
kind: IngressClass
metadata:
  name: alb
  annotations:
    ingressclass.kubernetes.io/is-default-class: "true"
spec:
  controller: eks.amazonaws.com/alb
```

Terapkan IngressClass ke cluster Anda:

```
kubectl apply -f ingressclass.yaml
```

Menyebarkan aplikasi sampel game 2048

Di bagian ini, kami memandu Anda melalui langkah-langkah untuk menyebarkan “game 2048” yang populer sebagai contoh aplikasi di dalam cluster. Manifes yang disediakan mencakup anotasi khusus untuk Application Load Balancer (ALB). Anotasi ini terintegrasi dengan dan menginstruksikan EKS untuk menangani lalu lintas HTTP yang masuk sebagai “menghadap ke internet” dan merutekannya ke layanan yang sesuai di `game-2048` namespace menggunakan tipe target “ip”.

Note

`docker-2048` Gambar dalam contoh adalah gambar `x86_64` kontainer dan tidak akan berjalan pada arsitektur lain.

1. Buat namespace Kubernetes yang dipanggil dengan bendera. `game-2048 --save-config`

```
kubectl create namespace game-2048 --save-config
```

Anda akan melihat output respons berikut:

```
namespace/game-2048 created
```

2. Menyebarkan aplikasi [Contoh Game 2048](#).

```
kubectl apply -n game-2048 -f https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.8.0/docs/examples/2048/2048_full.yaml
```

Manifes ini menyiapkan Deployment, Service, dan Ingress Kubernetes untuk game-2048 namespace, menciptakan sumber daya yang diperlukan untuk menyebarkan dan mengekspos aplikasi di dalam kluster. game-2048 Ini mencakup pembuatan layanan bernama service-2048 yang mengekspos penyebaran pada port80, dan sumber daya Ingress bernama yang mendefinisikan aturan routing untuk lalu lintas HTTP ingress-2048 yang masuk dan anotasi untuk Application Load Balancer (ALB) yang menghadap ke internet. Anda akan melihat output respons berikut:

```
namespace/game-2048 configured
deployment.apps/deployment-2048 created
service/service-2048 created
ingress.networking.k8s.io/ingress-2048 created
```

3. Jalankan perintah berikut untuk mendapatkan sumber daya Ingress untuk game-2048 namespace.

```
kubectl get ingress -n game-2048
```

Anda akan melihat output respons berikut:

NAME	CLASS	HOSTS	ADDRESS
		PORTS	AGE
ingress-2048	alb	*	k8s-game2048-ingress2-eb379a0f83-378466616.region-code.elb.amazonaws.com
		80	31s

Anda harus menunggu beberapa menit untuk Application Load Balancer (ALB) untuk menyediakan sebelum Anda memulai langkah-langkah berikut.

4. Buka browser web dan masukkan ADDRESS dari langkah sebelumnya untuk mengakses aplikasi web. Misalnya:

```
k8s-game2048-ingress2-eb379a0f83-378466616.region-code.elb.amazonaws.com
```

Anda akan melihat game 2048 di browser Anda. Mainkan!

2048

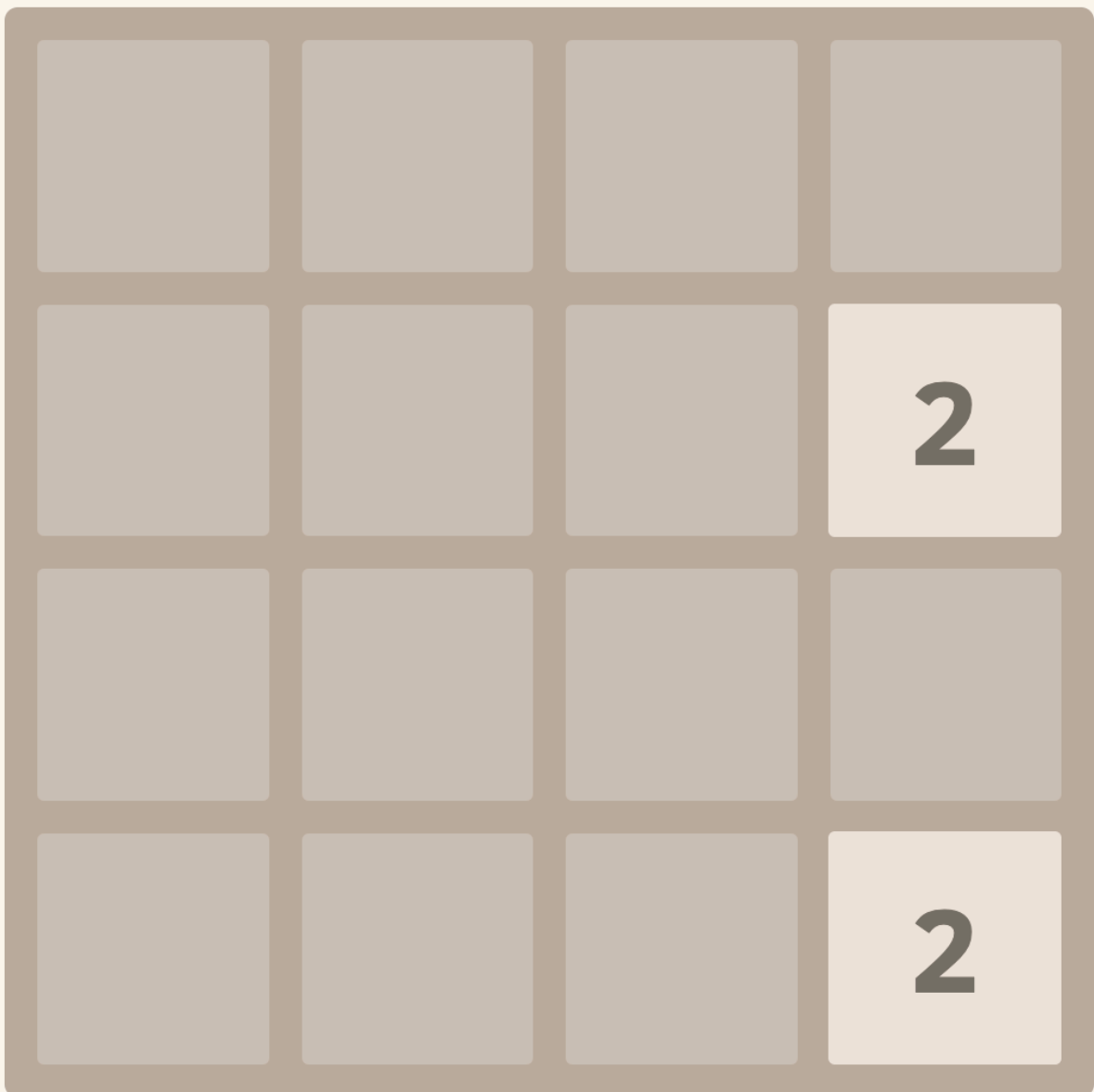
SCORE

0

BEST

48

Join the numbers and get to the **2048** tile!

[New Game](#)

Pertahankan Data menggunakan Mode Otomatis Amazon EKS

Sekarang game 2048 sudah aktif dan berjalan di cluster Amazon EKS Anda, saatnya untuk memastikan bahwa data game Anda disimpan dengan aman menggunakan kemampuan penyimpanan blok Amazon EKS Auto Mode.

1. Buat file bernama `storage-class.yaml`:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: auto-ebs-sc
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
provisioner: ebs.csi.eks.amazonaws.com
volumeBindingMode: WaitForFirstConsumer
parameters:
  type: gp3
  encrypted: "true"
```

2. Terapkan `StorageClass`:

```
kubectl apply -f storage-class.yaml
```

3. Buat Klaim Volume Persisten (PVC) untuk meminta penyimpanan data game Anda. Buat file bernama `ebs-pvc.yaml` dan tambahkan konten berikut ke dalamnya:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: game-data-pvc
  namespace: game-2048
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: auto-ebs-sc
```

4. Terapkan PVC ke cluster Anda:

```
kubectl apply -f ebs-pvc.yaml
```

Anda akan melihat output respons berikut:

```
persistentvolumeclaim/game-data-pvc created
```

5. Sekarang, Anda perlu memperbarui penerapan game 2048 Anda untuk menggunakan PVC ini untuk menyimpan data. Penyebaran berikut dikonfigurasi untuk menggunakan PVC untuk menyimpan data game. Buat file bernama `ebs-deployment.yaml` dan tambahkan konten berikut ke dalamnya:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: game-2048
  name: deployment-2048
spec:
  replicas: 3 # Adjust the number of replicas as needed
  selector:
    matchLabels:
      app.kubernetes.io/name: app-2048
  template:
    metadata:
      labels:
        app.kubernetes.io/name: app-2048
    spec:
      containers:
        - name: app-2048
          image: public.ecr.aws/l6m2t8p7/docker-2048:latest
          imagePullPolicy: Always
          ports:
            - containerPort: 80
          volumeMounts:
            - name: game-data
              mountPath: /var/lib/2048
      volumes:
        - name: game-data
          persistentVolumeClaim:
            claimName: game-data-pvc
```

6. Terapkan penerapan yang diperbarui:

```
kubectl apply -f ebs-deployment.yaml
```

Anda akan melihat output respons berikut:

```
deployment.apps/deployment-2048 configured
```

Dengan langkah-langkah ini, game 2048 Anda di cluster sekarang diatur untuk mempertahankan data menggunakan kemampuan penyimpanan blok Amazon EKS Auto Mode. Ini memastikan bahwa kemajuan dan data game Anda aman bahkan jika terjadi kegagalan pod atau node.

Jika Anda menyukai tutorial ini, beri tahu kami dengan memberikan umpan balik sehingga kami dapat memberi Anda lebih banyak tutorial mulai cepat khusus kasus penggunaan seperti ini.

Hapus cluster dan node Anda

Setelah Anda selesai dengan cluster yang Anda buat untuk tutorial ini, Anda harus membersihkan dengan menghapus cluster dengan perintah berikut. Jika Anda ingin melakukan lebih banyak hal dengan cluster ini sebelum membersihkan, lihat Langkah selanjutnya.

```
eksctl delete cluster -f ./cluster-config.yaml
```

EKS akan secara otomatis membersihkan node apa pun yang disediakan saat cluster dihapus.

Pelajari Amazon EKS dengan contoh

Gambaran Umum

Panduan Pengguna Amazon EKS ini berisi prosedur tujuan umum untuk membuat kluster EKS pertama Anda dari [baris perintah](#) atau [AWS Management Console](#) dan referensi yang solid untuk semua komponen Amazon EKS utama. Namun, sebagai administrator atau pengembang kluster Amazon EKS, Anda dapat memperoleh pemahaman yang lebih dalam tentang Amazon EKS dengan mengikuti jalur pembelajaran yang ada di situs di luar panduan ini. Situs-situs ini dapat membantu Anda:

- Siapkan jenis cluster tertentu. Jenis kluster tertentu dapat didasarkan pada jenis beban kerja atau persyaratan keamanan Anda. Misalnya, Anda mungkin ingin menyetel kluster untuk menjalankan beban kerja batch, machine learning, atau komputasi intensif.
- Tingkatkan cluster Anda. Anda dapat menambahkan fitur-fitur canggih ke cluster Anda untuk menyediakan hal-hal seperti observabilitas, penyimpanan fleksibel, penskalaan otomatis, atau jaringan cluster khusus.
- Otomatiskan pembaruan. Dengan menggunakan fitur seperti GitOps, Anda dapat mengatur untuk menyediakan infrastruktur kluster dan beban kerja secara otomatis, berdasarkan perubahan yang terjadi pada komponen tersebut di repositori Git Anda.
- Gunakan alat pengaturan cluster tingkat lanjut. Meskipun `eksctl` menyediakan cara cepat untuk membuat cluster, ada alat lain yang dapat membuatnya lebih mudah untuk mengkonfigurasi dan meningkatkan cluster yang lebih kompleks. Ini termasuk alat seperti [Terraform](#) dan [CloudFormation](#).

Untuk memulai jalur pembelajaran Amazon EKS Anda, saya sarankan Anda mengunjungi beberapa situs yang dijelaskan di halaman ini. Jika Anda mengalami masalah di sepanjang jalan, ada juga sumber daya untuk membantu Anda melewatinya. Misalnya, [Pusat Pengetahuan re:Post](#) memungkinkan Anda mencari database dukungan untuk masalah dukungan terkait Amazon EKS. Juga [Panduan Praktik Terbaik Amazon EKS](#) menawarkan tips tentang cara terbaik untuk menyiapkan kluster tingkat produksi Anda.

Lokakarya Amazon EKS

Dimulai dengan pemahaman dasar tentang Kubernetes dan container, [Amazon EKS workshop](#) adalah platform pembelajaran untuk memandu administrator cluster melalui fitur-fitur penting Amazon EKS. Berikut adalah cara Anda dapat terlibat dengan lokakarya Amazon EKS:

- **Dasar-Dasar Amazon EKS:** Tonton video di halaman [Pendahuluan](#) untuk mempelajari cara Amazon EKS mengimplementasikan fitur Kubernetes di cloud. AWS Jika Anda membutuhkan pemahaman yang lebih mendasar tentang Kubernetes, tonton video [What is](#) Kubernetes.
- **Penyiapan Amazon EKS:** Jika Anda memiliki AWS akun, bagian [Pengaturan](#) membantu Anda menyiapkan CloudShell lingkungan untuk membuat klaster. Ini menawarkan pilihan [eksctl](#) (baris perintah pembuatan cluster sederhana) dan [Terraform](#) (infrastructure-as-code pendekatan yang lebih untuk membuat cluster) untuk membuat cluster Amazon EKS Anda.
- **Amazon EKS Memulai:** Cobalah toko web sederhana dari bagian [aplikasi Contoh](#). Anda dapat menggunakan ini di seluruh latihan lainnya. Di bagian ini, Anda juga dapat mempelajari tentang [packaging image container](#) dan bagaimana microservices dikelola menggunakan Kubernetes Pods, Deployments, Services, dan Namespaces. StatefulSets Kemudian gunakan Kustomize untuk menerapkan perubahan pada manifes Kubernetes.
- **Amazon EKS Fundamentals:** Menggunakan AWS fitur-fitur seperti [AWS Load Balancer](#) Controller, lokakarya ini menunjukkan cara mengekspos aplikasi Anda ke dunia luar. Untuk penyimpanan, bengkel ini menampilkan cara menggunakan [Amazon EBS untuk penyimpanan blok](#), [Amazon EFS](#) untuk penyimpanan sistem file, dan Amazon untuk ONTAP FSx untuk mengelola sistem file NetApp ONTAP. AWS Untuk manajemen node, lokakarya membantu Anda mengatur [Grup Node Terkelola](#).
- **Fitur canggih Amazon EKS:** Fitur yang lebih canggih yang ditawarkan melalui bengkel Amazon EKS termasuk laboratorium untuk pengaturan:
 - [Autoscaling:](#) Ini termasuk penskalaan otomatis node (dengan Cluster Autoscaler atau Karpenter) dan penskalaan otomatis beban kerja (dengan Horizontal Pod Autoscaler dan Cluster Proportional Autoscaler).
 - [Observabilitas:](#) Pelajari tentang Logging, OpenSearch, Wawasan Kontainer di Amazon EKS, dan Visibilitas Biaya dengan Kubecost di satu set lab Observability.
 - **Keamanan:** Kumpulan [lab Keamanan](#) ini memungkinkan Anda menjelajahi [Secrets Management](#), [Amazon GuardDuty](#), [Pod Security Standards](#), dan manajemen [kebijakan Kyverno](#).

- Jaringan: Pelajari fitur jaringan untuk Amazon EKS dari lab [Networking](#) yang menyertakan [Amazon VPC CNI](#) (plugin jaringan pendukung) dan Amazon [VPC Lattice \(untuk mengonfigurasi cluster di seluruh VC dan akun pengguna\)](#).
- Otomatisasi: Labs on [Automation](#) mengarahkan Anda melalui [GitOps](#) metode pengelolaan cluster dan proyek Anda seperti [AWS Controllers for Kubernetes](#) dan Crossplane [untuk](#) mengelola pesawat kontrol Amazon EKS.

Tutorial penyiapan kluster langsung Amazon EKS

Satu set [tutorial Pengaturan Cluster Amazon EKS](#) di situs AWS Komunitas dapat membantu Anda membuat kluster Amazon EKS tujuan khusus dan meningkatkan kluster tersebut dengan berbagai cara. Tutorial dibagi menjadi tiga jenis:

Membangun cluster

Tutorial ini membantu Anda membangun cluster yang dapat digunakan untuk tujuan khusus. Tujuan khusus ini termasuk kemampuan untuk menjalankan:

- [Aplikasi yang dapat diskalakan secara global berdasarkan IPv6](#)
- [Tugas batch asinkron](#)
- [Layanan mikro lalu lintas tinggi](#)
- [Autoscaling dengan Karpenter di Fargate](#)
- [Beban kerja keuangan](#)
- [Grup Node Terkelola Windows](#)

Meningkatkan cluster

Setelah Anda memiliki kluster yang ada, Anda dapat memperluas dan meningkatkan kluster itu dengan cara yang memungkinkannya menjalankan beban kerja khusus dan sebaliknya meningkatkan cluster. Tutorial ini mencakup cara-cara untuk:

- [Menyediakan solusi penyimpanan dengan EFS CSI](#)
- [Menyediakan penyimpanan database dinamis dengan EBS CSI](#)
- [Mengekspos aplikasi pada IPv4 cluster menggunakan Load AWS Balancer Controller](#)
- [Mengekspos aplikasi pada IPv6 cluster menggunakan Load AWS Balancer Controller](#)

Mengoptimalkan layanan AWS

Dengan menggunakan tutorial ini, Anda dapat mengintegrasikan cluster Anda dengan AWS layanan dengan lebih baik. Tutorial ini termasuk yang membantu Anda:

- [Kelola catatan DNS untuk layanan mikro dengan ExternalDNS](#)
- [Memantau aplikasi dengan CloudWatch](#)
- [Kelola tugas asinkron dengan penyimpanan SQS dan EFS](#)
- [Konsumsi AWS Secrets Manager Secrets dari beban kerja](#)
- [Siapkan mTL dengan Fargate, NGINX, dan ACM PCA](#)

Sampel Amazon EKS

Toko repositori [Amazon EKS Sampel](#) bermanifestasi untuk digunakan dengan Amazon EKS. Manifestasi ini memberi Anda kesempatan untuk mencoba berbagai jenis aplikasi di Amazon EKS atau membuat jenis kluster Amazon EKS tertentu. Sampel termasuk manifes untuk:

- [Buat klaster AWS Amazon EKS Fargate](#)
- [Buat cluster dengan peran IAM yang ada](#)
- [Tambahkan dan Grup Node Terkelola Ubuntu ke cluster](#)
- [Cadangkan dan pulihkan penyimpanan Pod dengan snapshot volume](#)
- [Pulihkan volume EBS yang dipasang seperti PVCs beberapa akun](#)
- [Aktifkan protokol proxy untuk NGINX Ingress Controller dengan Classic Load Balancers](#)
- [Konfigurasi Logging di Fargate ke AWS OpenSearch](#)
- [Jalankan Python SDK dengan penyedia identitas federasi web](#)
- [Menerapkan aplikasi sampel pada pengontrol NFS CSI](#)
- [Gunakan snapshot volume untuk StatefulSets](#)
- [Menerapkan pod di seluruh node pada zona ketersediaan yang berbeda](#)

Perlu diingat bahwa sampel ini hanya untuk tujuan pembelajaran dan pengujian dan tidak dimaksudkan untuk digunakan dalam produksi.

AWS Tutorial

Situs [AWS Tutorial](#) menerbitkan beberapa tutorial Amazon EKS, tetapi juga menawarkan alat pencarian untuk menemukan tutorial lain yang diterbitkan di AWS situs (seperti situs AWS Komunitas). Tutorial Amazon EKS yang diterbitkan langsung di situs ini meliputi:

- [Menerapkan Aplikasi Web Kontainer di Amazon EKS](#)
- [Jalankan kluster Kubernetes dengan harga lebih murah \(Amazon EKS dan instans Spot\)](#)
- [Cara mengoptimalkan biaya pekerjaan Jenkins di Kubernetes](#)

Lokakarya Pengembang

Jika Anda seorang pengembang perangkat lunak, ingin membuat atau memfaktorkan ulang aplikasi untuk dijalankan di Amazon EKS, [lokakarya Pengembang Amazon EKS](#) adalah tempat yang baik untuk memulai. Lokakarya ini tidak hanya membantu Anda membangun aplikasi kontainer, tetapi juga membantu Anda menyebarkan kontainer tersebut ke registri kontainer ([ECR](#)) dan dari sana ke kluster Amazon EKS.

Mulailah dengan [Amazon EKS Python Workshop](#) untuk melalui proses refactoring aplikasi python, lalu atur lingkungan pengembangan Anda untuk mempersiapkan penerapan aplikasi. Melangkah melalui bagian-bagian pada Container, Kubernetes, dan Amazon EKS untuk mempersiapkan menjalankan aplikasi kontainer Anda di lingkungan tersebut.

Lokakarya Terraform

Meskipun `eksctl` merupakan alat sederhana untuk membuat cluster, untuk infrastructure-as-code jenis penyebaran Amazon EKS yang lebih kompleks, [Terraform](#) adalah alat pembuatan dan manajemen kluster Amazon EKS yang populer. [Workshop Terraform Amazon EKS](#) mengajarkan cara menggunakan Terraform untuk membuat VPC AWS, membuat kluster Amazon EKS, dan menambahkan penyempurnaan opsional ke cluster Anda. Secara khusus, ada bagian untuk membuat [cluster Amazon EKS pribadi](#)

AWS Pelatihan Amazon EKS

AWS menawarkan pelatihan formal untuk belajar tentang Amazon EKS. Kursus pelatihan tiga hari berjudul [Running Containers on Amazon Elastic Kubernetes Service](#) mengajarkan:

- Dasar-dasar Kubernetes dan Amazon EKS
- Cara membangun kluster Amazon EKS
- Mengamankan Amazon EKS dengan otorisasi AWS IAM dan Kubernetes RBAC
- GitOps alat otomatisasi
- Alat pemantauan
- Teknik untuk meningkatkan biaya, efisiensi, dan ketahanan

Memulai dengan Amazon EKS

Pastikan Anda siap untuk menggunakan Amazon EKS sebelum melalui panduan memulai. Untuk informasi selengkapnya, lihat [Penyiapan](#).

Ada dua panduan memulai yang tersedia untuk membuat kluster Kubernetes baru dengan simpul di Amazon EKS:

- [Memulai Amazon EKS — eksctl](#) — Panduan memulai ini membantu Anda menginstal semua sumber daya yang diperlukan untuk memulai menggunakan Amazon EKS dengan `eksctl`, utilitas baris perintah sederhana untuk membuat dan mengelola cluster Kubernetes di Amazon EKS. Di akhir tutorial, Anda akan memiliki kluster Amazon EKS yang sudah berjalan yang dapat digunakan sebagai tempat men-deploy aplikasi. Ini adalah cara tercepat dan paling sederhana untuk memulai dengan Amazon EKS.
- [Memulai Amazon EKS — AWS Management Console dan AWS CLI](#) — Panduan memulai ini membantu Anda membuat semua sumber daya yang diperlukan untuk memulai Amazon EKS menggunakan dan AWS Management Console dan CLI AWS. Di akhir tutorial, Anda akan memiliki kluster Amazon EKS yang sudah berjalan yang dapat digunakan sebagai tempat men-deploy aplikasi. Dalam panduan ini, Anda membuat setiap sumber daya yang diperlukan untuk kluster Amazon EKS secara manual. Prosedur ini memberikan visibilitas tentang bagaimana setiap sumber daya dibuat dan bagaimana mereka berinteraksi satu sama lain.

Kami juga menawarkan referensi berikut:

- Untuk koleksi tutorial langsung, lihat [EKS Cluster Setup](#) on AWS Community.
- Untuk contoh kode, lihat [Contoh kode untuk Amazon EKS menggunakan AWS SDKs](#).

Memulai Amazon EKS — Mode Otomatis EKS

Seperti pengalaman memulai EKS lainnya, membuat cluster pertama Anda dengan Mode Otomatis EKS mendelegasikan pengelolaan cluster itu sendiri. Namun, Mode Otomatis EKS memperluas otomatisasi EKS dengan menyerahkan tanggung jawab atas banyak layanan penting yang diperlukan untuk menyiapkan infrastruktur beban kerja (node, jaringan, dan berbagai layanan), sehingga lebih mudah untuk mengelola node dan meningkatkan skala untuk memenuhi tuntutan beban kerja.

Pilih salah satu cara berikut untuk membuat cluster dengan Mode Otomatis EKS:

- [the section called “ AWS CLI”](#): Gunakan antarmuka baris aws perintah untuk membuat cluster.
- [the section called “Konsol manajemen”](#): Gunakan AWS Management Console untuk membuat cluster.
- [the section called “CLI eksctl”](#): Gunakan antarmuka baris eksctl perintah untuk membuat cluster.

Jika Anda membandingkan berbagai pendekatan untuk membuat kluster EKS pertama Anda, Anda harus tahu bahwa Mode Otomatis EKS telah AWS mengambil alih tanggung jawab manajemen kluster tambahan yang mencakup pengaturan komponen untuk:

- Mulai dan skala node saat permintaan beban kerja meningkat dan menurun.
- Secara teratur meng-upgrade cluster itu sendiri (control plane), sistem operasi node, dan layanan yang berjalan pada node.
- Pilih pengaturan default yang menentukan hal-hal seperti ukuran dan kecepatan penyimpanan node dan konfigurasi jaringan Pod.

Untuk detail tentang apa yang Anda dapatkan dengan kluster Mode Otomatis EKS, lihat [Mode Otomatis EKS](#).

Memulai dengan Amazon EKS - **eksctl**

Note

Topik ini mencakup memulai tanpa Mode Otomatis EKS. Mode Otomatis EKS mengotomatiskan tugas rutin untuk komputasi cluster, penyimpanan, dan jaringan. [Pelajari cara memulai dengan Amazon EKS Auto Mode](#).

Panduan ini membantu Anda membuat semua sumber daya yang diperlukan untuk memulai Amazon Elastic Kubernetes Service (Amazon EKS) menggunakan eksctl, yaitu sebuah utilitas baris perintah sederhana untuk membuat dan mengelola kluster Kubernetes di Amazon EKS. Di akhir tutorial, Anda akan memiliki kluster Amazon EKS yang sudah berjalan yang dapat digunakan sebagai tempat men-deploy aplikasi.

Prosedur dalam panduan ini membuatkan beberapa sumber daya untuk Anda secara otomatis, yang seharusnya secara manual, ketika Anda membuat kluster menggunakan AWS Management Console. Jika Anda lebih suka membuat sebagian besar sumber daya secara manual untuk lebih memahami

bagaimana mereka berinteraksi satu sama lain, gunakan AWS Management Console untuk membuat cluster dan komputasi Anda. Untuk informasi selengkapnya, lihat [the section called “Buat cluster \(Konsol dan CLI\)”](#).

Prasyarat

[Sebelum memulai tutorial ini, Anda harus menginstal dan mengkonfigurasi alat AWS CLI, kubectl, dan eksctl seperti yang dijelaskan dalam Mengatur untuk menggunakan Amazon EKS.](#)

Langkah 1: Buat klaster dan simpul Amazon EKS Anda

Important

Untuk memulai sesederhana dan secepat mungkin, topik ini mencakup langkah-langkah untuk membuat klaster dan simpul dengan pengaturan default. Sebelum membuat klaster dan simpul untuk produksi, kami merekomendasikan supaya Anda membiasakan diri dengan semua pengaturan, dan men-deploy klaster serta simpul dengan pengaturan yang sesuai dengan kebutuhan Anda. Untuk informasi selengkapnya, silakan lihat [the section called “Membuat klaster”](#) dan [Kelola komputasi](#). Beberapa pengaturan hanya dapat diaktifkan saat membuat cluster dan node Anda.

Anda dapat membuat sebuah klaster dengan salah satu jenis simpul berikut. Untuk mempelajari selengkapnya tentang setiap jenis, lihat [Kelola komputasi](#). Setelah klaster Anda di-deploy, Anda dapat menambahkan jenis simpul lainnya.

- Fargate — Linux — Pilih jenis node ini jika Anda ingin menjalankan aplikasi Linux. [the section called “AWS Fargate”](#) Fargate adalah mesin komputasi tanpa server yang memungkinkan Anda menerapkan Pod Kubernetes tanpa mengelola instans Amazon. EC2
- Node terkelola - Linux - Pilih jenis node ini jika Anda ingin menjalankan aplikasi Amazon Linux di EC2 instans Amazon. Meskipun tidak tercantum dalam panduan ini, Anda juga dapat menambahkan simpul [Windows swakelola](#) dan [Bottlerocket](#) ke klaster Anda.

Buat cluster Amazon EKS Anda dengan perintah berikut. Anda dapat mengganti *my-cluster* dengan nilai Anda sendiri. Nama hanya dapat berisi karakter alfanumerik (peka huruf besar/kecil) dan tanda hubung. Itu harus dimulai dengan karakter alfanumerik dan tidak boleh lebih dari 100 karakter. Nama harus unik di dalam AWS Wilayah dan AWS akun tempat Anda membuat klaster.

Ganti *region-code* dengan AWS Wilayah apa pun yang didukung oleh Amazon EKS. Untuk daftar AWS Wilayah, lihat [titik akhir dan kuota Amazon EKS di panduan](#) Referensi AWS Umum.

Example

Fargate - Linux

```
eksctl create cluster --name my-cluster --region region-code --fargate
```

Managed nodes - Linux

```
eksctl create cluster --name my-cluster --region region-code
```

Pembuatan cluster membutuhkan waktu beberapa menit. Selama pembuatan Anda akan melihat beberapa baris output. Baris terakhir output mirip dengan baris contoh berikut.

```
[...]
[#] EKS cluster "my-cluster" in "region-code" region is ready
```

eksctl membuat file kubectl konfigurasi di `~/.kube/config` atau menambahkan konfigurasi cluster baru dalam file konfigurasi yang ada `~/.kube/config` di komputer Anda.

Setelah pembuatan klaster selesai, lihat AWS CloudFormation tumpukan bernama `eksctl-my-cluster` di AWS CloudFormation [konsol](#) untuk melihat semua sumber daya yang dibuat.

Langkah 2: Lihat sumber daya Kubernetes

1. Lihat simpul klaster Anda.

```
kubectl get nodes -o wide
```

Contoh output adalah sebagai berikut.

Example

Fargate - Linux

NAME			STATUS	ROLES	AGE
VERSION	INTERNAL-IP	EXTERNAL-IP	OS-IMAGE		KERNEL-VERSION
	CONTAINER-RUNTIME				

```
fargate-ip-192-0-2-0.region-code.compute.internal    Ready    <none>
8m3s    v1.2.3-eks-1234567    192.0.2.0    <none>    Amazon Linux 2
1.23.456-789.012.amzn2.x86_64    containerd://1.2.3
fargate-ip-192-0-2-1.region-code.compute.internal    Ready    <none>
7m30s    v1.2.3-eks-1234567    192-0-2-1    <none>    Amazon Linux 2
1.23.456-789.012.amzn2.x86_64    containerd://1.2.3
```

Managed nodes - Linux

NAME	INTERNAL-IP	EXTERNAL-IP	OS-IMAGE	STATUS	ROLES	AGE	VERSION
CONTAINER-RUNTIME							
ip-192-0-2-0.region-code.compute.internal				Ready	<none>	6m7s	
v1.2.3-eks-1234567	192.0.2.0		192.0.2.2		Amazon Linux 2		
1.23.456-789.012.amzn2.x86_64			containerd://1.2.3				
ip-192-0-2-1.region-code.compute.internal				Ready	<none>	6m4s	
v1.2.3-eks-1234567	192.0.2.1		192.0.2.3		Amazon Linux 2		
1.23.456-789.012.amzn2.x86_64			containerd://1.2.3				

Untuk informasi lebih lanjut tentang apa yang Anda lihat di output, lihat [the section called “Akses sumber daya kluster”](#).

2. Melihat beban kerja yang berjalan di kluster Anda.

```
kubectl get pods -A -o wide
```

Contoh output adalah sebagai berikut.

Example

Fargate - Linux

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE	IP
NODE		NOMINATED NODE				
READINESS	GATES					
kube-system	coredns-1234567890-abcde	1/1	Running	0	18m	
192.0.2.0	fargate-ip-192-0-2-0.region-code.compute.internal				<none>	
<none>						
kube-system	coredns-1234567890-12345	1/1	Running	0	18m	
192.0.2.1	fargate-ip-192-0-2-1.region-code.compute.internal				<none>	
<none>						

Managed nodes - Linux

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE	IP
	NODE			NOMINATED NODE	READINESS	
GATES						
kube-system	aws-node-12345	1/1	Running	0	7m43s	
192.0.2.1	ip-192-0-2-1.region-code.compute.internal			<none>		<none>
kube-system	aws-node-67890	1/1	Running	0	7m46s	
192.0.2.0	ip-192-0-2-0.region-code.compute.internal			<none>		<none>
kube-system	coredns-1234567890-abcde	1/1	Running	0	14m	
192.0.2.3	ip-192-0-2-3.region-code.compute.internal			<none>		<none>
kube-system	coredns-1234567890-12345	1/1	Running	0	14m	
192.0.2.4	ip-192-0-2-4.region-code.compute.internal			<none>		<none>
kube-system	kube-proxy-12345	1/1	Running	0	7m46s	
192.0.2.0	ip-192-0-2-0.region-code.compute.internal			<none>		<none>
kube-system	kube-proxy-67890	1/1	Running	0	7m43s	
192.0.2.1	ip-192-0-2-1.region-code.compute.internal			<none>		<none>

Untuk informasi lebih lanjut tentang apa yang Anda lihat di output, lihat [the section called “Akses sumber daya kluster”](#).

Langkah 3: Hapus kluster dan simpul Anda

Setelah Anda selesai dengan cluster dan node yang Anda buat untuk tutorial ini, Anda harus membersihkan dengan menghapus cluster dan node dengan perintah berikut. Jika Anda ingin melakukan lebih banyak dengan kluster ini sebelum Anda membersihkannya, lihat [the section called “Langkah selanjutnya”](#).

```
eksctl delete cluster --name my-cluster --region region-code
```

Langkah selanjutnya

Topik dokumentasi berikut membantu Anda untuk memperluas fungsionalitas kluster Anda.

- Menerapkan [aplikasi sampel](#) ke cluster Anda.
- [Prinsipal IAM](#) yang membuat kluster adalah satu-satunya prinsipal yang dapat melakukan panggilan ke server API Kubernetes dengan atau. `kubectl` AWS Management Console Jika Anda ingin prinsipal IAM lainnya memiliki akses ke cluster Anda, maka Anda perlu menambahkannya.

Untuk informasi selengkapnya, silakan lihat [the section called “Akses API Kubernetes”](#) dan [the section called “Izin yang diperlukan”](#).

- [Sebelum menerapkan cluster untuk penggunaan produksi, sebaiknya Anda membiasakan diri dengan semua pengaturan untuk cluster dan node.](#) Beberapa pengaturan (seperti mengaktifkan akses SSH ke EC2 node Amazon) harus dilakukan saat cluster dibuat.
- Untuk meningkatkan keamanan klaster Anda, [konfigurasi plugin Amazon VPC Container Networking Interface untuk menggunakan peran IAM untuk](#) akun layanan.

Memulai Amazon EKS — AWS Management Console dan AWS CLI

Note

Topik ini mencakup memulai tanpa Mode Otomatis EKS. Ini menggunakan Grup Node Terkelola untuk menyebarkan node.

Mode Otomatis EKS mengotomatiskan tugas rutin untuk komputasi cluster, penyimpanan, dan jaringan. [Pelajari cara memulai dengan Amazon EKS Auto Mode.](#) EKS Auto Mode adalah metode yang disukai untuk menyebarkan node.

Panduan ini membantu Anda membuat semua sumber daya yang diperlukan untuk memulai Amazon Elastic Kubernetes Service (Amazon EKS) menggunakan AWS Management Console dan CLI. AWS Dalam panduan ini, Anda membuat setiap sumber daya secara manual. Di akhir tutorial, Anda akan memiliki klaster Amazon EKS yang sudah berjalan yang dapat digunakan sebagai tempat men-deploy aplikasi.

Prosedur dalam panduan ini memberikan visibilitas lengkap tentang bagaimana setiap sumber daya dibuat dan bagaimana mereka berinteraksi satu sama lain. Jika Anda lebih suka memiliki sebagian besar sumber daya yang dibuat untuk Anda secara otomatis, gunakan `eksctl` CLI untuk membuat cluster dan node Anda. Untuk informasi selengkapnya, lihat [the section called “Buat cluster \(eksctl\)”](#).

Prasyarat

Sebelum memulai tutorial ini, Anda harus menginstal dan mengonfigurasi alat-alat dan sumber daya yang Anda butuhkan berikut untuk membuat dan mengelola sebuah klaster Amazon EKS.

- **AWS CLI** — Alat baris perintah untuk bekerja dengan AWS layanan, termasuk Amazon EKS. Untuk informasi selengkapnya, lihat [Menginstal](#) di Panduan Pengguna Antarmuka Baris AWS Perintah. Setelah menginstal AWS CLI, kami sarankan Anda juga mengkonfigurasinya. Untuk informasi selengkapnya, lihat [Konfigurasi cepat dengan aws configure](#) di Panduan Pengguna Antarmuka Baris AWS Perintah. Perhatikan bahwa AWS CLI v2 diperlukan untuk menggunakan opsi update-kubeconfig yang ditampilkan di halaman ini.
- **kubectl** — Alat baris perintah untuk bekerja dengan cluster Kubernetes. Untuk informasi selengkapnya, lihat [the section called “Siapkan kubectl dan eksctl”](#).
- **Izin IAM yang diperlukan** - Prinsip keamanan IAM yang Anda gunakan harus memiliki izin untuk bekerja dengan peran Amazon EKS IAM, peran terkait layanan, VPC, dan sumber AWS CloudFormation daya terkait. Untuk informasi selengkapnya, lihat [Tindakan](#) dan [Menggunakan peran terkait layanan](#) di Panduan Pengguna IAM. Anda harus menyelesaikan semua langkah dalam panduan ini sebagai pengguna yang sama. Untuk memeriksa pengguna saat ini, jalankan perintah berikut:

```
aws sts get-caller-identity
```

Kami menyarankan Anda menyelesaikan langkah-langkah dalam topik ini di shell Bash. Jika Anda tidak menggunakan shell Bash, beberapa perintah skrip seperti karakter kelanjutan baris dan cara variabel diatur dan digunakan memerlukan penyesuaian untuk shell Anda. Selain itu, aturan mengutip dan melarikan diri untuk shell Anda mungkin berbeda. Untuk informasi selengkapnya, lihat [Menggunakan tanda kutip dengan string di AWS CLI di](#) Panduan Pengguna Antarmuka Baris AWS Perintah.

Langkah 1: Buat klaster dan Amazon EKS Anda

Important

Untuk memulai sesederhana dan secepat mungkin, topik ini mencakup langkah-langkah untuk membuat cluster dengan pengaturan default. Sebelum membuat cluster untuk penggunaan produksi, kami sarankan Anda membiasakan diri dengan semua pengaturan dan menyebarkan cluster dengan pengaturan yang memenuhi persyaratan Anda. Untuk informasi selengkapnya, lihat [the section called “Membuat klaster”](#). Beberapa pengaturan hanya dapat diaktifkan saat membuat cluster Anda.

1. Buat Amazon VPC dengan subnet publik dan privat yang memenuhi persyaratan Amazon EKS. Ganti *region-code* dengan AWS Wilayah apa pun yang didukung oleh Amazon EKS. Untuk daftar AWS Wilayah, lihat [titik akhir dan kuota Amazon EKS di panduan Referensi AWS Umum](#). Anda dapat mengganti *my-eks-vpc-stack* dengan nama apa pun yang Anda pilih.

```
aws cloudformation create-stack \  
  --region region-code \  
  --stack-name my-eks-vpc-stack \  
  --template-url https://s3.us-west-2.amazonaws.com/amazon-eks/  
cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml
```

Tip

Untuk daftar semua sumber daya yang dibuat oleh perintah sebelumnya, buka AWS CloudFormation konsol di <https://console.aws.amazon.com/cloudformation/>. Pilih *my-eks-vpc-stack* tumpukan dan kemudian pilih tab Sumber Daya.

2. Membuat IAM role kluster dan lampirkan kebijakan terkelola Amazon EKS IAM ke dalam peran itu. Cluster Kubernetes yang dikelola oleh Amazon EKS melakukan panggilan ke AWS layanan lain atas nama Anda untuk mengelola sumber daya yang Anda gunakan dengan layanan.
 - a. Salin isi berikut ke file bernama *eks-cluster-role-trust-policy.json*.

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Principal": {  
        "Service": "eks.amazonaws.com"  
      },  
      "Action": "sts:AssumeRole"  
    }  
  ]  
}
```

- b. Buat peran.

```
aws iam create-role \  
  --role-name myAmazonEKSClusterRole \  
  --policy-arn arn:aws:iam::aws:policy/AmazonEKS_FullAccess
```

```
--assume-role-policy-document file://"eks-cluster-role-trust-policy.json"
```

- c. Lampirkan kebijakan terkelola IAM Amazon EKS yang diperlukan untuk peran tersebut.

```
aws iam attach-role-policy \  
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy \  
  --role-name myAmazonEKSClusterRole
```

3. Buka konsol Amazon EKS di <https://console.aws.amazon.com/eks/home#/clusters>.

Pastikan bahwa AWS Wilayah yang ditampilkan di kanan atas konsol Anda adalah AWS Wilayah tempat Anda ingin membuat kluster. Jika tidak, pilih dropdown di sebelah nama AWS Region dan pilih AWS Region yang ingin Anda gunakan.

4. Pilih Buat kluster. Jika Anda tidak melihat opsi ini, pilih Cluster di panel navigasi kiri terlebih dahulu.
5. Pada halaman Configure cluster, lakukan hal berikut:
 - a. Pilih Konfigurasi khusus dan nonaktifkan Gunakan Mode Otomatis EKS. (Jika Anda lebih suka kluster Mode Otomatis EKS, lihat saja [the section called “Konsol manajemen”](#).)
 - b. Masukkan Nama untuk kluster Anda, seperti *my-cluster*. Nama hanya dapat berisi karakter alfanumerik (peka huruf besar/kecil) dan tanda hubung. Itu harus dimulai dengan karakter alfanumerik dan tidak boleh lebih dari 100 karakter. Nama harus unik di dalam AWS Wilayah dan AWS akun tempat Anda membuat cluster.
 - c. Untuk Peran Layanan Cluster, pilih *myAmazonEKSClusterRole*.
 - d. Biarkan pengaturan yang tersisa pada nilai default mereka dan pilih Berikutnya.
6. Pada halaman Tentukan jaringan, lakukan hal berikut:
 - a. Pilih ID VPC yang Anda buat pada langkah sebelumnya dari daftar dropdown VPC. Ini adalah sesuatu seperti ** / my-eks-vpc-stack-VPC*.
 - b. Pilih subnet yang dibuat pada langkah sebelumnya dari daftar dropdown Subnet. Subnet akan menjadi sesuatu seperti ** / my-eks-vpc-stack-**.
 - c. Pilih grup keamanan yang dibuat pada langkah sebelumnya dari daftar dropdown grup keamanan tambahan. Ini adalah sesuatu seperti ** / my-eks-vpc-stack-ControlPlaneSecurityGroup-**.
 - d. Biarkan pengaturan yang tersisa pada nilai default mereka dan pilih Berikutnya.
7. Pada halaman Konfigurasi observabilitas, pilih Berikutnya.

8. Pada halaman Pilih add-on, pilih Berikutnya.

Untuk informasi lebih lanjut tentang add-on, lihat [the section called “Add-on Amazon EKS”](#).

9. Pada halaman Konfigurasi pengaturan add-on yang dipilih, pilih Berikutnya.

10. Pada halaman Tinjau dan buat, pilih Buat.

Di sebelah kanan nama cluster, status cluster adalah Creating selama beberapa menit hingga proses penyediaan cluster selesai. Jangan lanjutkan ke langkah berikutnya sampai statusnya Aktif.

 Note

Anda mungkin menerima kesalahan bahwa salah satu Availability Zone dalam permintaan Anda tidak memiliki kapasitas yang cukup untuk membuat klaster Amazon EKS. Jika hal ini terjadi, output galat berisi Availability Zones yang dapat mendukung klaster baru. Cobalah untuk kembali membuat klaster dengan setidaknya dua subnet yang terletak di Availability Zones yang didukung untuk akun Anda. Untuk informasi selengkapnya, lihat [the section called “Kapasitas tidak mencukupi”](#).

Langkah 2: Konfigurasi komputer Anda agar bisa berkomunikasi dengan klaster Anda

Di bagian ini, Anda akan membuat file kubeconfig untuk klaster Anda. Pengaturan dalam file ini mengaktifkan kubectl CLI agar dapat berkomunikasi dengan klaster Anda.

Sebelum melanjutkan, pastikan pembuatan klaster Anda berhasil diselesaikan di Langkah 1.

1. Buat atau perbarui file kubeconfig untuk klaster Anda. Ganti *region-code* dengan AWS Region tempat Anda membuat cluster. Ganti *my-cluster* dengan nama klaster Anda.

```
aws eks update-kubeconfig --region region-code --name my-cluster
```

Secara default, config file dibuat ~/.kube atau konfigurasi cluster baru ditambahkan ke config file yang ada di ~/.kube.

2. Uji konfigurasi Anda.

```
kubectl get svc
```

 Note

Jika Anda menerima kesalahan otorisasi atau jenis sumber daya, lihat [the section called “Tidak sah atau akses ditolak \(kubectl\)”](#) di topik pemecahan masalah.

Contoh output adalah sebagai berikut.

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
svc/kubernetes	ClusterIP	10.100.0.1	<none>	443/TCP	1m

Langkah 3: Buat node

 Important

Untuk memulai sesederhana dan secepat mungkin, topik ini mencakup langkah-langkah untuk membuat node dengan sebagian besar pengaturan default. Sebelum membuat node untuk penggunaan produksi, kami sarankan Anda membiasakan diri dengan semua pengaturan dan menyebarkan node dengan pengaturan yang memenuhi persyaratan Anda. Untuk informasi selengkapnya, lihat [Kelola komputasi](#). Beberapa pengaturan hanya dapat diaktifkan saat membuat node Anda.

Prosedur ini mengonfigurasi kluster Anda untuk menggunakan grup node Terkelola untuk membuat node, menentukan subnet dan peran IAM node yang Anda buat pada langkah sebelumnya. Ini memungkinkan Anda menjalankan aplikasi Amazon Linux di EC2 instans Amazon.

Untuk mempelajari lebih lanjut tentang berbagai cara untuk mengkonfigurasi node di EKS, lihat [Kelola komputasi](#). Setelah kluster Anda di-deploy, Anda dapat menambahkan jenis simpul lainnya. Meskipun tidak tercantum dalam panduan ini, Anda juga dapat menambahkan simpul [Windows swakelola](#) dan [Bottlerocket](#) ke kluster Anda.

Untuk membuat grup node terkelola EC2 Linux

1. Buat IAM role simpul dan lampirkan kebijakan terkelola Amazon EKS IAM ke dalam peran itu. kubeletDaemon simpul Amazon EKS melakukan panggilan ke AWS APIs atas nama Anda. Simpul menerima izin untuk panggilan API ini melalui profil instans IAM dan kebijakan terkait.

- a. Salin konten berikut ke file bernama `node-role-trust-policy.json`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

- b. Buat IAM role simpel.

```
aws iam create-role \
  --role-name myAmazonEKSNodeRole \
  --assume-role-policy-document file://"node-role-trust-policy.json"
```

- c. Lampirkan kebijakan IAM terkelola yang diperlukan ke peran.

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name myAmazonEKSNodeRole
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name myAmazonEKSNodeRole
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name myAmazonEKSNodeRole
```

- d. Buka konsol Amazon EKS di <https://console.aws.amazon.com/eks/home#/clusters>.
- e. Pilih nama cluster yang Anda buat di [Langkah 1: Buat klaster Amazon EKS Anda](#), seperti *my-cluster*.
- f. Pada *my-cluster* halaman, lakukan hal berikut:
- g. Pilih tab Compute.
- h. Pilih Tambahkan Grup Node.

2. Pada halaman Configure Node Group, lakukan hal berikut:

- a. Untuk Nama, masukkan nama unik untuk grup node terkelola Anda, seperti *my-nodegroup*. Nama grup node tidak boleh lebih dari 63 karakter. Itu harus dimulai dengan huruf atau digit, tetapi juga dapat menyertakan tanda hubung dan garis bawah untuk karakter yang tersisa.
 - b. Untuk nama peran IAM Node, pilih *myAmazonEKSTaskRole* peran yang Anda buat di langkah sebelumnya. Kami merekomendasikan bahwa setiap grup node menggunakan peran IAM uniknya sendiri.
 - c. Pilih Berikutnya.
3. Pada halaman Setel konfigurasi komputasi dan penskalaan, terima nilai default dan pilih Berikutnya.
 4. Pada halaman Tentukan jaringan, terima nilai default dan pilih Berikutnya.
 5. Pada halaman Tinjauan dan buat, tinjau konfigurasi grup simpul terkelola dan pilih Buat.
 6. Setelah beberapa menit, Status di bagian Konfigurasi Grup simpul akan berubah dari Sedang membuat ke Aktif. Jangan lanjutkan ke langkah berikutnya sampai statusnya Aktif.

Langkah 4: Lihat sumber daya

Anda dapat melihat simpul dan beban kerja Kubernetes Anda.

1. Pada panel navigasi sebelah kiri, pilih Klaster. Dalam daftar Cluster, pilih nama cluster yang Anda buat, seperti *my-cluster*.
2. Pada *my-cluster* halaman, pilih yang berikut ini:
 - a. Tab komputasi — Anda melihat daftar Node yang digunakan untuk cluster. Anda dapat memilih nama node untuk melihat informasi lebih lanjut tentangnya.
 - b. Tab Resources — Anda melihat semua resource Kubernetes yang di-deploy secara default ke klaster Amazon EKS. Pilih jenis sumber daya apa pun di konsol untuk mempelajarinya lebih lanjut.

Langkah 5: Hapus sumber daya

Setelah Anda selesai dengan cluster dan node yang Anda buat untuk tutorial ini, Anda harus menghapus sumber daya yang Anda buat. Jika Anda ingin melakukan lebih banyak hal dengan cluster ini sebelum menghapus sumber daya, lihat [the section called “Langkah selanjutnya”](#).

1. Hapus profil grup node apa pun yang Anda buat.
 - a. Buka konsol Amazon EKS di <https://console.aws.amazon.com/eks/home#/clusters>.

- b. Pada panel navigasi sebelah kiri, pilih Klaster. Dalam daftar cluster, pilih **my-cluster**.
 - c. Pilih tab Compute.
 - d. Jika Anda membuat grup node, pilih grup **my-nodegroup** node dan kemudian pilih Hapus. Masukkan **my-nodegroup**, lalu pilih Hapus.
 - e. Jangan lanjutkan sampai profil grup node dihapus.
2. Hapus klaster .
- a. Pada panel navigasi sebelah kiri, pilih Klaster. Dalam daftar cluster, pilih **my-cluster**.
 - b. Pilih Hapus klaster.
 - c. Masukkan **my-cluster** dan kemudian pilih Hapus. Jangan lanjutkan sampai cluster dihapus.
3. Hapus AWS CloudFormation tumpukan VPC yang Anda buat.
- a. Buka CloudFormation konsol di <https://console.aws.amazon.com/cloudformation/>.
 - b. Pilih **my-eks-vpc-stack** tumpukan, lalu pilih Hapus.
 - c. Di kotak dialog Hapus **my-eks-vpc-stack** konfirmasi, pilih Hapus tumpukan.
4. Hapus IAM role yang Anda buat.
- a. Buka konsol IAM di <https://console.aws.amazon.com/iam/>.
 - b. Di panel navigasi sebelah kiri, pilih Peran.
 - c. Pilih setiap peran yang Anda buat dari daftar (**myAmazonEKSClusterRole**, serta **myAmazonEKSNodeRole**). Pilih Hapus, masukkan teks konfirmasi yang diminta, lalu pilih Hapus.

Langkah selanjutnya

Topik dokumentasi berikut membantu Anda untuk memperluas fungsionalitas klaster Anda.

- [Prinsipal IAM](#) yang membuat cluster adalah satu-satunya prinsipal yang dapat melakukan panggilan ke server API Kubernetes dengan atau. kubectl AWS Management Console Jika Anda ingin prinsipal IAM lainnya memiliki akses ke cluster Anda, maka Anda perlu menambahkannya. Untuk informasi selengkapnya, lihat [the section called “Akses API Kubernetes”](#) dan [the section called “Izin yang diperlukan”](#).
- Menerapkan [aplikasi sampel](#) ke cluster Anda.
- [Sebelum menerapkan cluster untuk penggunaan produksi, sebaiknya Anda membiasakan diri dengan semua pengaturan untuk cluster dan node.](#) Beberapa pengaturan (seperti mengaktifkan akses SSH ke EC2 node Amazon) harus dilakukan saat cluster dibuat.

- Untuk meningkatkan keamanan klaster Anda, [konfigurasi plugin Amazon VPC Container Networking Interface untuk menggunakan peran IAM untuk](#) akun layanan.

Mengotomatiskan infrastruktur kluster dengan Mode Otomatis EKS

EKS Auto Mode memperluas AWS pengelolaan kluster Kubernetes di luar kluster itu sendiri, AWS untuk memungkinkan juga menyiapkan dan mengelola infrastruktur yang memungkinkan kelancaran pengoperasian beban kerja Anda. Anda dapat mendelegasikan keputusan infrastruktur utama dan memanfaatkan keahlian AWS untuk day-to-day operasi. Infrastruktur kluster yang dikelola oleh AWS mencakup banyak kemampuan Kubernetes sebagai komponen inti, sebagai lawan dari add-on, seperti compute autoscaling, pod dan service networking, application load balancing, cluster DNS, block storage, dan dukungan GPU.

Untuk memulai, Anda dapat menerapkan kluster Mode Otomatis EKS baru atau mengaktifkan Mode Otomatis EKS pada cluster yang ada. Anda dapat menerapkan, meningkatkan, atau memodifikasi kluster Mode Otomatis EKS Anda menggunakan eksctl, AWS CLI, EKS, AWS Management Console atau alat pilihan Anda. APIs infrastructure-as-code

Dengan Mode Otomatis EKS, Anda dapat terus menggunakan alat yang kompatibel dengan Kubernetes pilihan Anda. Mode Otomatis EKS terintegrasi dengan AWS layanan seperti Amazon EC2, Amazon EBS, dan ELB, memanfaatkan sumber daya AWS cloud yang mengikuti praktik terbaik. Sumber daya ini secara otomatis diskalakan, dioptimalkan biaya, dan diperbarui secara berkala untuk membantu meminimalkan biaya operasional dan overhead.

Fitur

Mode Otomatis EKS menyediakan fitur tingkat tinggi berikut:

Merampingkan Manajemen Cluster Kubernetes: Mode Otomatis EKS merampingkan manajemen EKS dengan menyediakan kluster siap produksi dengan overhead operasional minimal. Dengan Mode Otomatis EKS, Anda dapat menjalankan beban kerja dinamis yang menuntut dengan percaya diri, tanpa memerlukan keahlian EKS yang mendalam.

Ketersediaan Aplikasi: Mode Otomatis EKS secara dinamis menambahkan atau menghapus node di kluster EKS Anda berdasarkan permintaan aplikasi Kubernetes Anda. Ini meminimalkan kebutuhan akan perencanaan kapasitas manual dan memastikan ketersediaan aplikasi.

Efisiensi: Mode Otomatis EKS dirancang untuk mengoptimalkan biaya komputasi sambil mematuhi fleksibilitas yang ditentukan oleh persyaratan Anda NodePool dan beban kerja. Ini juga

menghentikan instance yang tidak digunakan dan mengkonsolidasikan beban kerja ke node lain untuk meningkatkan efisiensi biaya.

Keamanan: Penggunaan Mode Otomatis EKS AMIs yang diperlakukan sebagai tidak dapat diubah untuk node Anda. Ini AMIs memberlakukan perangkat lunak terkunci, mengaktifkan kontrol akses SELinux wajib, dan menyediakan sistem file root read-only. Selain itu, node yang diluncurkan oleh EKS Auto Mode memiliki masa pakai maksimum 21 hari (yang dapat Anda kurangi), setelah itu mereka secara otomatis diganti dengan node baru. Pendekatan ini meningkatkan postur keamanan Anda dengan bersepeda node secara teratur, menyelaraskan dengan praktik terbaik yang telah diadopsi oleh banyak pelanggan.

Upgrade Otomatis: Mode Otomatis EKS membuat kluster Kubernetes, node, dan komponen terkait tetap up to date dengan patch terbaru, sembari menghormati Pod Disruption Budgets () dan Disruption Budgets () yang sudah dikonfigurasi. PDBs NodePool NDBs Hingga masa pakai maksimum 21 hari, intervensi mungkin diperlukan jika pemblokiran PDBs atau konfigurasi lain mencegah pembaruan.

Komponen Terkelola: Mode Otomatis EKS menyertakan Kubernetes dan fitur AWS cloud sebagai komponen inti yang seharusnya harus dikelola sebagai add-on. Ini termasuk dukungan bawaan untuk penetapan alamat IP Pod, kebijakan jaringan Pod, layanan DNS lokal, plug-in GPU, pemeriksa kesehatan, dan penyimpanan EBS CSI.

Dapat disesuaikan NodePools dan NodeClasses: Jika beban kerja Anda memerlukan perubahan pada konfigurasi penyimpanan, komputasi, atau jaringan, Anda dapat membuat kustom NodePools dan NodeClasses menggunakan Mode Otomatis EKS. Meskipun Anda tidak boleh mengedit default NodePools dan NodeClasses, Anda dapat menambahkan kustom baru NodePools atau NodeClasses di samping konfigurasi default untuk memenuhi persyaratan spesifik Anda.

Komponen Otomatis

Mode Otomatis EKS merampingkan pengoperasian kluster Amazon EKS Anda dengan mengotomatiskan komponen infrastruktur utama. Mengaktifkan Mode Otomatis EKS semakin mengurangi tugas untuk mengelola kluster EKS Anda.

Berikut ini adalah daftar komponen bidang data yang otomatis:

- **Komputasi:** Untuk banyak beban kerja, dengan Mode Otomatis EKS Anda dapat melupakan banyak aspek komputasi untuk kluster EKS Anda. Ini termasuk:

- **Node:** Node Mode Otomatis EKS dirancang untuk diperlakukan seperti peralatan. Mode Otomatis EKS melakukan hal berikut:
 - Memilih AMI yang sesuai yang dikonfigurasi dengan banyak layanan yang diperlukan untuk menjalankan beban kerja Anda tanpa intervensi.
 - Mengunci akses ke file di AMI menggunakan mode SELinux penegakan dan sistem file root read-only.
 - Mencegah akses langsung ke node dengan melarang akses SSH atau SSM.
 - Termasuk dukungan GPU, dengan driver dan plugin kernel terpisah untuk NVIDIA dan Neuron GPUs, memungkinkan beban kerja berkinerja tinggi.
 - Secara otomatis menangani [pemberitahuan interupsi Instans EC2 Spot](#) dan EC2 kejadian kesehatan Instance
- **Penskalaan otomatis:** Mengandalkan [penskalaan otomatis Karpenter](#), Mode Otomatis EKS memantau Pod yang tidak dapat dijadwalkan dan memungkinkan node baru digunakan untuk menjalankan pod tersebut. Saat beban kerja dihentikan, Mode Otomatis EKS secara dinamis mengganggu dan menghentikan node saat tidak lagi diperlukan, mengoptimalkan penggunaan sumber daya.
- **Upgrade:** Mengambil kendali node Anda merampingkan kemampuan EKS Auto Mode untuk menyediakan patch keamanan dan sistem operasi dan upgrade komponen sesuai kebutuhan. Upgrade tersebut dirancang untuk memberikan gangguan minimal pada beban kerja Anda. Mode Otomatis EKS memberlakukan masa pakai node maksimum 21 hari untuk memastikan up-to-date perangkat lunak dan APIs
- **Load balancing:** EKS Auto Mode merampingkan load balancing dengan mengintegrasikan layanan Elastic Load Balancing Amazon, mengotomatiskan penyediaan dan konfigurasi load balancer untuk Layanan Kubernetes dan sumber daya Ingress. Ini mendukung fitur-fitur canggih untuk Application dan Network Load Balancer, mengelola siklus hidup mereka, dan menskalakannya agar sesuai dengan permintaan klaster. Integrasi ini menyediakan solusi penyeimbangan beban siap produksi yang mengikuti praktik AWS terbaik, memungkinkan Anda untuk fokus pada aplikasi daripada manajemen infrastruktur.
- **Penyimpanan:** Mode Otomatis EKS mengonfigurasi penyimpanan sementara untuk Anda dengan menyiapkan jenis volume, ukuran volume, kebijakan enkripsi, dan kebijakan penghapusan setelah penghentian node.
- **Networking:** EKS Auto Mode mengotomatiskan tugas-tugas jaringan penting untuk Pod dan konektivitas layanan. Ini termasuk IPv4/IPv6 dukungan dan penggunaan blok CIDR sekunder untuk memperluas ruang alamat IP.

- Identity and Access Management: Anda tidak perlu menginstal EKS Pod Identity Agent pada kluster Mode Otomatis EKS.

Untuk informasi lebih lanjut tentang komponen ini, lihat [the section called “Cara kerjanya”](#).

Konfigurasi

Meskipun Mode Otomatis EKS akan secara efektif mengelola sebagian besar layanan pesawat data Anda tanpa campur tangan Anda, mungkin ada saat-saat ketika Anda ingin mengubah perilaku beberapa layanan tersebut. Anda dapat memodifikasi konfigurasi kluster Mode Otomatis EKS Anda dengan cara berikut:

- Kubernetes DaemonSets: Daripada memodifikasi layanan yang diinstal pada node Anda, Anda dapat menggunakan daemonset Kubernetes. Daemonset dirancang untuk dikelola oleh Kubernetes, tetapi berjalan di setiap node di cluster. Dengan cara ini, Anda dapat menambahkan layanan khusus untuk memantau atau mengawasi node Anda.
- Kustom NodePools dan NodeClasses: Default NodePools dan NodeClasses dikonfigurasi oleh Mode Otomatis EKS dan Anda tidak boleh mengeditnya. Untuk menyesuaikan perilaku node, Anda dapat membuat tambahan NodePools atau NodeClasses untuk kasus penggunaan seperti:
 - Memilih jenis instans tertentu (misalnya, prosesor yang dipercepat atau instance EC2 Spot).
 - Mengisolasi beban kerja untuk tujuan keamanan atau pelacakan biaya.
 - Mengkonfigurasi pengaturan penyimpanan sementara seperti IOPS, ukuran, dan throughput.
- Load Balancing: Beberapa layanan, seperti load balancing, yang dijalankan EKS Auto Mode sebagai objek Kubernetes, dapat dikonfigurasi langsung pada kluster Mode Otomatis EKS Anda.

Untuk informasi selengkapnya tentang opsi untuk mengonfigurasi Mode Otomatis EKS, lihat [the section called “Konfigurasi”](#).

Buat cluster dengan Amazon EKS Auto Mode

Bab ini menjelaskan cara membuat cluster Amazon EKS dengan Mode Otomatis diaktifkan menggunakan berbagai alat dan antarmuka. Mode Otomatis menyederhanakan pembuatan kluster dengan mengonfigurasi dan mengelola infrastruktur komputasi, jaringan, dan penyimpanan cluster secara otomatis. Anda akan belajar cara membuat cluster Mode Otomatis menggunakan AWS CLI, AWS Management Console, atau alat baris perintah eksctl.

 Note

Mode Otomatis EKS membutuhkan Kubernetes versi 1.29 atau lebih tinggi. Mode Otomatis EKS tidak tersedia di `ap-southeast-7` atau `mx-central-1` AWS Wilayah.

Pilih alat pilihan Anda berdasarkan kebutuhan Anda: AWS Management Console Ini menyediakan antarmuka visual yang ideal untuk mempelajari fitur Mode Otomatis EKS dan membuat cluster individual. AWS CLI paling cocok untuk tugas skrip dan otomatisasi, terutama saat mengintegrasikan pembuatan kluster ke dalam alur kerja atau saluran CI/CD yang ada. CLI `eksctl` menawarkan pengalaman asli Kubernetes dan direkomendasikan bagi pengguna yang akrab dengan perkakas Kubernetes yang menginginkan operasi baris perintah yang disederhanakan dengan default yang masuk akal.

Sebelum memulai, pastikan Anda memiliki prasyarat yang diperlukan diinstal dan dikonfigurasi, termasuk izin IAM yang sesuai untuk membuat kluster EKS. Untuk mempelajari cara menginstal alat CLI seperti `kubectl`, dan `aws eksctl`, lihat. [Penyiapan](#)

Anda dapat menggunakan AWS CLI, AWS Management Console, atau `eksctl` CLI untuk membuat cluster dengan Amazon EKS Auto Mode.

Topik

- [Buat Kluster Mode Otomatis EKS dengan CLI `eksctl`](#)
- [Buat Kluster Mode Otomatis EKS dengan AWS CLI](#)
- [Buat Kluster Mode Otomatis EKS dengan AWS Management Console](#)

Buat Kluster Mode Otomatis EKS dengan CLI `eksctl`

Topik ini menunjukkan cara membuat cluster Amazon EKS Auto Mode menggunakan antarmuka baris perintah `eksctl` (CLI). Anda dapat membuat kluster Mode Otomatis baik dengan menjalankan satu perintah CLI atau dengan menerapkan file konfigurasi YAML. Kedua metode menyediakan fungsionalitas yang sama, dengan pendekatan YAML yang menawarkan kontrol yang lebih terperinci atas pengaturan cluster.

CLI `eksctl` menyederhanakan proses pembuatan dan pengelolaan kluster Mode Otomatis EKS dengan menangani pembuatan dan konfigurasi sumber daya yang mendasarinya. AWS Sebelum melanjutkan, pastikan Anda memiliki AWS kredensi dan izin yang diperlukan yang dikonfigurasi pada

mesin lokal Anda. Panduan ini mengasumsikan Anda terbiasa dengan konsep dasar Amazon EKS dan telah menginstal alat CLI yang diperlukan.

Note

Anda harus menginstal versi 0.195.0 atau lebih besar dari eksctl. Untuk informasi lebih lanjut, lihat rilis [eksctl](#) di GitHub

Buat kluster Mode Otomatis EKS dengan perintah CLI

Anda harus memiliki aws dan eksctl alat yang diinstal. Anda harus masuk ke AWS CLI dengan izin yang cukup untuk mengelola AWS sumber daya termasuk: EC2 instance, EC2 jaringan, kluster EKS, dan peran IAM. Untuk informasi selengkapnya, lihat [Penyiapan](#).

Jalankan perintah berikut untuk membuat kluster Mode Otomatis EKS baru dengan

```
eksctl create cluster --name=<cluster-name> --enable-auto-mode
```

Buat kluster Mode Otomatis EKS dengan file YAMAL

Anda harus memiliki aws dan eksctl alat yang diinstal. Anda harus masuk ke AWS CLI dengan izin yang cukup untuk mengelola AWS sumber daya termasuk: EC2 instance, EC2 jaringan, kluster EKS, dan peran IAM. Untuk informasi selengkapnya, lihat [Penyiapan](#).

Tinjau opsi konfigurasi Mode Otomatis EKS dalam ClusterConfig sumber daya contoh di bawah ini. Untuk ClusterConfig spesifikasi lengkapnya, lihat dokumentasi [eksctl](#).

AWS menyarankan mengaktifkan Mode Otomatis EKS. Jika ini adalah pertama kalinya Anda membuat kluster Mode Otomatis EKS, biarkan yang `nodeRoleARN` tidak ditentukan untuk membuat Peran IAM Node untuk Mode Otomatis EKS. Jika Anda sudah memiliki Peran IAM Node di AWS akun Anda, AWS sarankan untuk menggunakannya kembali.

AWS menyarankan untuk tidak menentukan nilai apa pun untuk `nodePools`. Mode Otomatis EKS akan membuat kumpulan node default. Anda dapat menggunakan API Kubernetes untuk membuat kumpulan node tambahan.

```
# cluster.yaml
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
```



```
metadata:
  name: <cluster-name>
  region: <aws-region>

iam:
  # ARN of the Cluster IAM Role
  # optional, eksctl creates a new role if not supplied
  # suggested to use one Cluster IAM Role per account
  serviceRoleARN: <arn-cluster-iam-role>

autoModeConfig:
  # defaults to false
  enabled: boolean
  # optional, defaults to [general-purpose, system].
  # suggested to leave unspecified
  # To disable creation of nodePools, set it to the empty array ([]).
  nodePools: []string
  # optional, eksctl creates a new role if this is not supplied
  # and nodePools are present.
  nodeRoleARN: string
```

Simpan ClusterConfig file sebagaiccluster.yaml, dan gunakan perintah berikut untuk membuat cluster:

```
eksctl create cluster -f cluster.yaml
```

Buat Kluster Mode Otomatis EKS dengan AWS CLI

EKS Auto Mode Clusters mengotomatiskan tugas manajemen cluster rutin untuk komputasi, penyimpanan, dan jaringan. Misalnya, Kluster Mode Otomatis EKS secara otomatis mendeteksi kapan node tambahan diperlukan dan menyediakan EC2 instance baru untuk memenuhi tuntutan beban kerja.

Topik ini memandu Anda membuat Kluster Mode Otomatis EKS baru menggunakan AWS CLI dan secara opsional menerapkan contoh beban kerja.

Prasyarat

- Versi terbaru dari AWS Command Line Interface (AWS CLI) diinstal dan dikonfigurasi pada perangkat Anda. Untuk memeriksa versi Anda saat ini, gunakan `aws --version`. Untuk

menginstal versi terbaru, lihat [Menginstal](#) dan [Konfigurasi cepat](#) dengan aws configure di Panduan Pengguna Antarmuka Baris AWS Perintah.

- Masuk ke CLI dengan izin IAM yang cukup untuk membuat AWS sumber daya termasuk Kebijakan IAM, Peran IAM, dan Kluster EKS.
- Alat baris perintah kubectl diinstal pada perangkat Anda. AWS menyarankan Anda menggunakan versi kubectl yang sama dengan versi Kubernetes dari EKS Cluster Anda. Untuk menginstal atau meng-upgrade kubectl, lihat [the section called “Siapkan kubectl dan eksctl”](#)

Tentukan subnet VPC

Mode Otomatis Amazon EKS menyebarkan node ke subnet VPC. Saat membuat cluster EKS, Anda harus menentukan subnet VPC tempat node akan digunakan. Anda dapat menggunakan subnet VPC default di AWS akun Anda atau membuat VPC khusus untuk beban kerja kritis.

- AWS menyarankan untuk membuat VPC khusus untuk cluster Anda. Pelajari cara [the section called “Buat VPC”](#).
- Konsol EKS membantu menciptakan VPC baru. Pelajari cara [the section called “Konsol manajemen”](#).
- Atau, Anda dapat menggunakan VPC default akun Anda AWS . Gunakan instruksi berikut untuk menemukan Subnet IDs.

Untuk menemukan Subnet IDs dari VPC default Anda

Menggunakan AWS CLI:

1. Jalankan perintah berikut untuk membuat daftar VPC default dan subnetnya:

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=$(aws ec2 describe-vpcs
--query 'Vpcs[?IsDefault==`true`].VpcId' --output text)" --query 'Subnets[*].
{ID:SubnetId,AZ:AvailabilityZone}' --output table
```

2. Simpan output dan perhatikan Subnet IDs.

Contoh output:

```
-----
| DescribeSubnets |
-----
```

SubnetId	AvailabilityZone
-----	-----
subnet-012345678	us-west-2a
subnet-234567890	us-west-2b
subnet-345678901	us-west-2c
-----	-----

Peran IAM untuk Kluster Mode Otomatis EKS

Peran IAM Cluster

Mode Otomatis EKS memerlukan Peran IAM Cluster untuk melakukan tindakan di AWS akun Anda, seperti menyediakan instance baru. EC2 Anda harus membuat peran ini untuk memberikan EKS izin yang diperlukan. AWS merekomendasikan untuk melampirkan kebijakan AWS terkelola berikut ke Peran IAM Cluster:

- [EKSComputeKebijakan Amazon](#)
- [Amazon EKSBLOCK StoragePolicy](#)
- [Amazon EKSLoad BalancingPolicy](#)
- [EKSNetworkingKebijakan Amazon](#)
- [EKSClusterKebijakan Amazon](#)

Peran IAM Node

Saat Anda membuat kluster Mode Otomatis EKS, Anda menentukan Peran IAM Node. Ketika Mode Otomatis EKS membuat node untuk memproses beban kerja yang tertunda, setiap node EC2 instance baru diberi Peran IAM Node. Peran ini memungkinkan node untuk berkomunikasi dengan EKS tetapi umumnya tidak diakses oleh beban kerja yang berjalan pada node.

Jika Anda ingin memberikan izin untuk beban kerja yang berjalan pada sebuah node, gunakan EKS Pod Identity. Untuk informasi selengkapnya, lihat [the section called "Identitas Pod"](#).

Anda harus membuat peran ini dan melampirkan kebijakan AWS terkelola berikut:

- [Amazon EKSWorker NodeMinimalPolicy](#)
- [Amazon EC2 ContainerRegistryPullOnly](#)

peran tertaut layanan

Mode Otomatis EKS juga memerlukan Peran Tertaut Layanan, yang secara otomatis dibuat dan dikonfigurasi oleh. AWS Untuk informasi lebih lanjut, lihat [AWSServiceRoleForAmazonEKS](#).

Membuat Peran IAM Kluster Mode Otomatis EKS

Langkah 1: Buat Kebijakan Kepercayaan

Buat kebijakan kepercayaan yang memungkinkan layanan Amazon EKS untuk mengambil peran. Simpan kebijakan sebagai `trust-policy.json`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

Langkah 2: Buat Peran IAM

Gunakan kebijakan kepercayaan untuk membuat Peran IAM Cluster:

```
aws iam create-role \
  --role-name AmazonEKSAutoClusterRole \
  --assume-role-policy-document file://trust-policy.json
```

Langkah 3: Perhatikan Peran ARN

Ambil dan simpan ARN dari peran baru untuk digunakan dalam langkah-langkah selanjutnya:

```
aws iam get-role --role-name AmazonEKSAutoClusterRole --query "Role.Arn" --output text
```

Langkah 4: Lampirkan Kebijakan yang Diperlukan

Lampirkan kebijakan AWS terkelola berikut ke Peran IAM Cluster untuk memberikan izin yang diperlukan:

EKSClusterKebijakan Amazon:

```
aws iam attach-role-policy \  
  --role-name AmazonEKSAutoClusterRole \  
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSClusterPolicy
```

EKSComputeKebijakan Amazon:

```
aws iam attach-role-policy \  
  --role-name AmazonEKSAutoClusterRole \  
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSComputePolicy
```

Amazon EKSBLOCK StoragePolicy:

```
aws iam attach-role-policy \  
  --role-name AmazonEKSAutoClusterRole \  
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSBLOCKStoragePolicy
```

Amazon EKSLoad BalancingPolicy:

```
aws iam attach-role-policy \  
  --role-name AmazonEKSAutoClusterRole \  
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSLoadBalancingPolicy
```

EKSNetworkingKebijakan Amazon:

```
aws iam attach-role-policy \  
  --role-name AmazonEKSAutoClusterRole \  
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSNetworkingPolicy
```

Buat Peran IAM Node Mode Otomatis EKS

Langkah 1: Buat Kebijakan Kepercayaan

Buat kebijakan kepercayaan yang memungkinkan layanan Amazon EKS untuk mengambil peran. Simpan kebijakan sebagai `node-trust-policy.json`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Langkah 2: Buat Peran IAM Node

Gunakan `node-trust-policyfile.json` dari langkah sebelumnya untuk menentukan entitas mana yang dapat mengambil peran. Jalankan perintah berikut untuk membuat Peran IAM Node:

```
aws iam create-role \
  --role-name AmazonEKSAutoNodeRole \
  --assume-role-policy-document file://node-trust-policy.json
```

Langkah 3: Perhatikan Peran ARN

Setelah membuat peran, ambil dan simpan ARN dari Peran IAM Node. Anda akan membutuhkan ARN ini pada langkah selanjutnya. Gunakan perintah berikut untuk mendapatkan ARN:

```
aws iam get-role --role-name AmazonEKSAutoNodeRole --query "Role.Arn" --output text
```

Langkah 4: Lampirkan Kebijakan yang Diperlukan

Lampirkan kebijakan AWS terkelola berikut ke Peran IAM Node untuk memberikan izin yang diperlukan:

Amazon EKSWorker NodeMinimalPolicy:

```
aws iam attach-role-policy \
  --role-name AmazonEKSAutoNodeRole \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodeMinimalPolicy
```

Amazon EC2 ContainerRegistryPullOnly:

```
aws iam attach-role-policy \  
  --role-name AmazonEKSAutoNodeRole \  
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryPullOnly
```

Buat Kluster Mode Otomatis EKS

Gambaran Umum

Untuk membuat Kluster Mode Otomatis EKS menggunakan AWS CLI, Anda memerlukan parameter berikut:

- `cluster-name`: Nama cluster.
- `k8s-version`: Versi Kubernetes (mis., 1.31).
- `subnet-ids`: Subnet IDs diidentifikasi pada langkah sebelumnya.
- `cluster-role-arn`: ARN dari Peran IAM Cluster.
- `node-role-arn`: ARN dari Peran IAM Node.

Konfigurasi Cluster Default

Tinjau nilai dan fitur default ini sebelum membuat cluster:

- `nodePools`: Mode Otomatis EKS mencakup Pools Node standar tujuan umum dan sistem. Pelajari lebih lanjut tentang [Node Pools](#).

Catatan: Kumpulan Node dalam Mode Otomatis EKS berbeda dari Grup Node Terkelola Amazon EKS tetapi dapat hidup berdampingan di cluster yang sama.

- `computeConfig.enabled`: Mengotomatiskan tugas komputasi rutin, seperti membuat dan menghapus EC2 instance.
- `kubernetesNetworkConfig.elasticLoadBalancing.enabled`: Mengotomatiskan tugas penyeimbangan beban, termasuk membuat dan menghapus Elastic Load Balancer.
- `storageConfig.blockStorage.enabled`: Mengotomatiskan tugas penyimpanan, seperti membuat dan menghapus volume Amazon EBS.
- `accessConfig.authenticationMode`: Memerlukan entri akses EKS. Pelajari lebih lanjut tentang [mode otentikasi EKS](#).

Jalankan Perintah

Gunakan perintah berikut untuk membuat cluster:

```
aws eks create-cluster \
  --region ${AWS_REGION} \
  --cli-input-json \
  "{
    \"name\": \"${CLUSTER_NAME}\",
    \"version\": \"${K8S_VERSION}\",
    \"roleArn\": \"${CLUSTER_ROLE_ARN}\",
    \"resourcesVpcConfig\": {
      \"subnetIds\": ${SUBNETS_JSON},
      \"endpointPublicAccess\": true,
      \"endpointPrivateAccess\": true
    },
    \"computeConfig\": {
      \"enabled\": true,
      \"nodeRoleArn\": \"${NODE_ROLE_ARN}\",
      \"nodePools\": [\"general-purpose\", \"system\"]
    },
    \"kubernetesNetworkConfig\": {
      \"elasticLoadBalancing\": {
        \"enabled\": true
      }
    },
    \"storageConfig\": {
      \"blockStorage\": {
        \"enabled\": true
      }
    },
    \"accessConfig\": {
      \"authenticationMode\": \"API\"
    }
  }
```

Periksa Status Cluster

Langkah 1: Verifikasi Pembuatan Cluster

Jalankan perintah berikut untuk memeriksa status cluster Anda. Pembuatan cluster biasanya memakan waktu sekitar 15 menit:


```
aws eks describe-cluster --name "${CLUSTER_NAME}" --output json
```

Langkah 2: Perbarui kubeconfig

Setelah cluster siap, perbarui file kubeconfig lokal Anda `kubectl` untuk mengaktifkan komunikasi dengan cluster. Konfigurasi ini menggunakan AWS CLI untuk otentikasi.

```
aws eks update-kubeconfig --name "${CLUSTER_NAME}"
```

Langkah 3: Verifikasi Node Pools

Buat daftar Node Pools di cluster Anda menggunakan perintah berikut:

```
kubectl get nodepools
```

Langkah Berikutnya

- Pelajari cara [menerapkan beban kerja sampel](#) ke kluster Mode Otomatis EKS baru Anda.

Buat Kluster Mode Otomatis EKS dengan AWS Management Console

Membuat kluster Mode Otomatis EKS di AWS Management Console memerlukan konfigurasi yang lebih sedikit daripada opsi lain. EKS terintegrasi dengan AWS IAM dan VPC Networking untuk membantu Anda membuat sumber daya yang terkait dengan cluster EKS.

Anda memiliki dua opsi untuk membuat cluster di konsol:

- Konfigurasi cepat (dengan Mode Otomatis EKS)
- Konfigurasi kustom

Dalam topik ini, Anda akan belajar cara membuat kluster Mode Otomatis EKS menggunakan opsi konfigurasi Cepat.

Buat Mode Otomatis EKS menggunakan opsi konfigurasi cepat

Anda harus masuk ke AWS Management Console dengan izin yang cukup untuk mengelola AWS sumber daya termasuk: EC2 instance, EC2 jaringan, kluster EKS, dan peran IAM.

1. Arahkan ke Konsol EKS

2. Klik Buat klaster
3. Konfirmasikan opsi konfigurasi Cepat dipilih
4. Tentukan nilai-nilai berikut, atau gunakan default untuk cluster uji.
 - Nama Cluster
 - Versi Kubernetes
5. Pilih Peran IAM Cluster. Jika ini adalah pertama kalinya Anda membuat kluster Mode Otomatis EKS, gunakan opsi Buat peran yang direkomendasikan.
 - Secara opsional, Anda dapat menggunakan kembali satu Peran IAM Cluster di AWS akun Anda untuk semua kluster Mode Otomatis EKS.
 - Peran IAM Cluster mencakup izin yang diperlukan untuk Mode Otomatis EKS untuk mengelola sumber daya termasuk EC2 instans, volume EBS, dan penyeimbang beban. EC2
 - Opsi Buat peran yang direkomendasikan mengisi semua bidang dengan nilai yang disarankan. Pilih Berikutnya dan kemudian Buat. Peran akan menggunakan `AmazonEKSAutoClusterRole` nama yang disarankan.
 - Jika Anda baru saja membuat peran baru, gunakan ikon Refresh untuk memuat ulang tarik-turun pemilihan peran.
6. Pilih Peran IAM Node. Jika ini adalah pertama kalinya Anda membuat kluster Mode Otomatis EKS, gunakan opsi Buat peran yang direkomendasikan.
 - Secara opsional, Anda dapat menggunakan kembali satu Peran IAM Node di AWS akun Anda untuk semua kluster Mode Otomatis EKS.
 - Peran IAM Node menyertakan izin yang diperlukan untuk node Mode Otomatis untuk terhubung ke cluster. Peran IAM Node harus menyertakan izin untuk mengambil gambar ECR untuk kontainer Anda.
 - Opsi Buat peran yang direkomendasikan mengisi semua bidang dengan nilai yang disarankan. Pilih Berikutnya dan kemudian Buat. Peran akan menggunakan `AmazonEKSAutoNodeRole` nama yang disarankan.
 - Jika Anda baru saja membuat peran baru, gunakan ikon Refresh untuk memuat ulang tarik-turun pemilihan peran.
7. Pilih VPC untuk kluster Mode Otomatis EKS Anda. Pilih Create VPC untuk membuat VPC baru untuk EKS, atau pilih VPC yang sebelumnya Anda buat untuk EKS.
 - Jika Anda menggunakan Konsol VPC untuk membuat VPC baru, AWS sarankan Anda membuat setidaknya satu Gateway NAT per Availability Zone. Jika tidak, Anda dapat menggunakan semua default lainnya

- Untuk informasi selengkapnya dan detail persyaratan IPv6 kluster, lihat [the section called “Buat VPC”](#).
8. (opsional) Mode Otomatis EKS secara otomatis mengisi subnet pribadi untuk VPC pilihan Anda. Anda dapat menghapus subnet yang tidak diinginkan.
- EKS secara otomatis memilih subnet pribadi dari VPC mengikuti praktik terbaik. Anda dapat memilih subnet tambahan dari VPC, seperti subnet publik.
9. (opsional) Pilih Lihat default konfigurasi cepat untuk meninjau semua nilai konfigurasi untuk cluster baru. Tabel menunjukkan beberapa nilai tidak dapat diedit setelah cluster dibuat.
10. Pilih Buat kluster. Perhatikan bahwa mungkin diperlukan waktu lima belas menit untuk menyelesaikan pembuatan kluster.

Langkah Berikutnya

- Pelajari cara [Menerapkan Beban Kerja Sampel ke kluster Mode Otomatis EKS](#)

Aktifkan Mode Otomatis EKS pada kluster EKS yang ada

Anda dapat mengaktifkan Mode Otomatis EKS pada Kluster EKS yang ada.

Note

Mode Otomatis EKS membutuhkan Kubernetes versi 1.29 atau lebih tinggi. Mode Otomatis EKS tidak tersedia di `ap-southeast-7` atau `mx-central-1` AWS Wilayah.

AWS mendukung migrasi berikut:

- Bermigrasi dari Karpenter ke node Mode Otomatis EKS. Untuk informasi selengkapnya, lihat [the section called “Migrasi dari Karpenter”](#).
- Migrasi dari Grup Node Terkelola EKS ke node Mode Otomatis EKS. Untuk informasi selengkapnya, lihat [the section called “Migrasi dari MNGs”](#).
- Bermigrasi dari EKS Fargate ke Mode Otomatis EKS. Untuk informasi selengkapnya, lihat [the section called “Migrasi dari Fargate”](#).

AWS tidak mendukung migrasi berikut:

- Migrasi volume dari pengontrol EBS CSI (menggunakan add-on Amazon EKS) ke pengontrol EKS Auto Mode EBS CSI (dikelola oleh Mode Otomatis EKS). PVCs dibuat dengan satu tidak dapat dipasang oleh yang lain, karena mereka menggunakan dua penyedia volume Kubernetes yang berbeda.
- Proyek [eks-auto-mode-ebs-migration-tool](#) (AWS Labs) memungkinkan migrasi antara standar EBS CSI StorageClass (`ebs.csi.aws.com`) dan EKS Auto EBS StorageClass CSI (`ebs.csi.eks.amazonaws.com`). Perhatikan bahwa migrasi memerlukan penghapusan dan pembuatan ulang PersistentVolumeClaim/PersistentVolume sumber daya yang ada, jadi validasi di lingkungan non-produksi sangat penting sebelum implementasi.

- Migrasi load balancer dari Load Balancer Controller ke AWS Mode Otomatis EKS

Anda dapat menginstal AWS Load Balancer Controller pada kluster Mode Otomatis Amazon EKS. Gunakan `loadBalancerClass` opsi `IngressClass` atau untuk mengaitkan sumber daya Service dan Ingress dengan Load Balancer Controller atau EKS Auto Mode.

- Migrasi kluster EKS dengan alternatif CNIs atau konfigurasi jaringan lain yang tidak didukung

Referensi migrasi

Gunakan referensi migrasi berikut untuk mengonfigurasi sumber daya Kubernetes yang akan dimiliki oleh pengontrol yang dikelola sendiri atau Mode Otomatis EKS.

Kemampuan	Sumber Daya	Bidang	Dikelola Sendiri	Mode Otomatis EKS
Penyimpanan blok	StorageClass	provisioner	<code>ebs.csi.aws.com</code>	<code>ebs.csi.eks.amazonaws.com</code>
Penyeimbangan beban	Service	loadBalancerClass	<code>service.k8s.aws/nlb</code>	<code>eks.amazonaws.com/nlb</code>
Penyeimbangan beban	IngressClass	controller	<code>ingress.k8s.aws/alb</code>	<code>eks.amazonaws.com/alb</code>

Kemampuan	Sumber Daya	Bidang	Dikelola Sendiri	Mode Otomatis EKS
Penyeimbangan beban	IngressClassParams	apiversion	elbv2.k8s.aws/v1beta1	eks.amazonaws.com/v1
Penyeimbangan beban	TargetGroupBinding	apiversion	elbv2.k8s.aws/v1beta1	eks.amazonaws.com/v1
Komputasi	NodeClass	apiVersion	karpenter.sh/v1	eks.amazonaws.com/v1

Migrasi volume EBS

Saat memigrasikan beban kerja ke Mode Otomatis EKS, Anda perlu menangani migrasi volume EBS karena penyedia driver CSI yang berbeda:

- Penyedia Mode Otomatis EKS: `ebs.csi.eks.amazonaws.com`
- Penyedia EBS CSI sumber terbuka: `ebs.csi.aws.com`

Ikuti langkah-langkah berikut untuk memigrasikan volume persisten Anda:

1. Ubah kebijakan retensi volume: Ubah versi platform (PV) yang ada `persistentVolumeReclaimPolicy Retain` untuk memastikan volume EBS yang mendasarinya tidak dihapus.
2. Hapus PV dari Kubernetes: Hapus sumber daya PV lama sambil menjaga volume EBS yang sebenarnya tetap utuh.
3. Buat PV baru dengan penyediaan statis: Buat PV baru yang mereferensikan volume EBS yang sama tetapi berfungsi dengan driver CSI target.
4. Mengikat ke PVC baru: Buat PVC baru yang secara khusus mereferensikan PV Anda menggunakan `volumeName` bidang.

Pertimbangan

- Pastikan aplikasi Anda dihentikan sebelum memulai migrasi ini.
- Cadangkan data Anda sebelum memulai proses migrasi.
- Proses ini perlu dilakukan untuk setiap volume persisten.
- Beban kerja harus diperbarui untuk menggunakan PVC baru.

Migrasi penyeimbang beban

Anda tidak dapat langsung mentransfer penyeimbang beban yang ada dari pengontrol penyeimbang AWS beban yang dikelola sendiri ke Mode Otomatis EKS. Sebagai gantinya, Anda harus menerapkan strategi penyebaran biru-hijau. Ini melibatkan mempertahankan konfigurasi penyeimbang beban yang ada saat membuat penyeimbang beban baru di bawah pengontrol terkelola.

Untuk meminimalkan gangguan layanan, kami merekomendasikan pendekatan perpindahan lalu lintas berbasis DNS. Pertama, buat penyeimbang beban baru dengan menggunakan Mode Otomatis EKS sambil menjaga konfigurasi yang ada tetap beroperasi. Kemudian, gunakan perutean DNS (seperti Route 53) untuk secara bertahap mengalihkan lalu lintas dari penyeimbang beban lama ke yang baru. Setelah lalu lintas berhasil dimigrasi dan Anda telah memverifikasi konfigurasi baru, Anda dapat menonaktifkan penyeimbang beban lama dan pengontrol yang dikelola sendiri.

Aktifkan Mode Otomatis EKS pada cluster yang ada

Topik ini menjelaskan cara mengaktifkan Mode Otomatis Amazon EKS di kluster Amazon EKS Anda yang ada. Mengaktifkan Mode Otomatis pada cluster yang ada memerlukan pembaruan izin IAM dan mengonfigurasi pengaturan Mode Otomatis EKS inti. Setelah diaktifkan, Anda dapat mulai memigrasikan beban kerja komputasi yang ada untuk memanfaatkan operasi Mode Otomatis yang disederhanakan dan manajemen infrastruktur otomatis.

Important

Pastikan Anda memiliki versi minimum yang diperlukan dari Add-on Amazon EKS tertentu yang diinstal sebelum mengaktifkan Mode Otomatis EKS. Untuk informasi selengkapnya, lihat [the section called “Versi add-on yang diperlukan”](#).

Sebelum memulai, pastikan Anda memiliki akses administrator ke kluster Amazon EKS dan izin untuk mengubah peran IAM. Langkah-langkah dalam topik ini memandu Anda melalui mengaktifkan Mode Otomatis menggunakan AWS CLI AWS Management Console atau.

AWS Management Console

Anda harus masuk ke AWS konsol dengan izin untuk mengelola IAM, EKS, dan EC2 sumber daya.

Note

Peran IAM Cluster dari EKS Cluster tidak dapat diubah setelah cluster dibuat. Mode Otomatis EKS memerlukan izin tambahan pada peran ini. Anda harus melampirkan kebijakan tambahan ke peran saat ini.

Perbarui peran IAM Cluster

1. Buka halaman ikhtisar cluster Anda di AWS Management Console.
2. Di bawah ARN peran IAM Cluster, pilih Lihat di IAM.
3. Dari tarik-turun Tambahkan Izin, pilih Lampirkan Kebijakan.
4. Gunakan kotak Pencarian untuk menemukan dan memilih kebijakan berikut:
 - AmazonEKSComputePolicy
 - AmazonEKSBlockStoragePolicy
 - AmazonEKSLoadBalancingPolicy
 - AmazonEKSNetworkingPolicy
 - AmazonEKSClusterPolicy
5. Pilih Tambahkan izin
6. Dari tab Hubungan kepercayaan, pilih Edit kebijakan kepercayaan
7. Masukkan kebijakan kepercayaan Peran IAM Cluster berikut, lalu pilih Perbarui kebijakan

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
```

```
    "Service": "eks.amazonaws.com"
  },
  "Action": [
    "sts:AssumeRole",
    "sts:TagSession"
  ]
}
```

Aktifkan Mode Otomatis EKS

1. Buka halaman ikhtisar cluster Anda di AWS Management Console.
2. Di bawah Mode Otomatis EKS pilih Kelola
3. Alihkan Mode Otomatis EKS ke aktif.
4. Dari dropdown EKS Node Pool, pilih kumpulan node default yang ingin Anda buat.
 - Pelajari lebih lanjut tentang Node Pools dalam Mode Otomatis EKS. Untuk informasi selengkapnya, lihat [the section called “Buat kumpulan simpul”](#).
5. Jika sebelumnya Anda telah membuat peran IAM Node Mode Otomatis EKS di AWS akun ini, pilih di dropdown Peran IAM Node. Jika Anda belum membuat peran ini sebelumnya, pilih Buat Peran yang direkomendasikan dan ikuti langkah-langkahnya.

AWS CLI

Prasyarat

- Peran IAM Cluster dari Kluster EKS yang ada harus menyertakan izin yang cukup untuk Mode Otomatis EKS, seperti kebijakan berikut:
 - AmazonEKSClusterPolicy
 - AmazonEKSBlockStoragePolicy
 - AmazonEKSLoadBalancingPolicy
 - AmazonEKSNetworkingPolicy
 - AmazonEKSClusterPolicy
- Peran IAM Cluster harus memiliki kebijakan kepercayaan yang diperbarui termasuk `sts:TagSession` tindakan. Untuk informasi selengkapnya tentang membuat Peran IAM Cluster, lihat [the section called “ AWS CLI”](#).

- awsCLI diinstal, masuk, dan versi yang cukup. Anda harus memiliki izin untuk mengelola IAM, EKS, dan EC2 sumber daya. Untuk informasi selengkapnya, lihat [Penyiapan](#).

Prosedur

Gunakan perintah berikut untuk mengaktifkan Mode Otomatis EKS pada cluster yang ada.

Note

Kemampuan komputasi, penyimpanan blok, dan penyeimbangan beban semuanya harus diaktifkan atau dinonaktifkan dalam permintaan yang sama.

```
aws eks update-cluster-config \
  --name $CLUSTER_NAME \
  --compute-config enabled=true \
  --kubernetes-network-config '{"elasticLoadBalancing":{"enabled": true}}' \
  --storage-config '{"blockStorage":{"enabled": true}}'
```

Versi add-on yang diperlukan

Jika Anda berencana untuk mengaktifkan Mode Otomatis EKS pada klaster yang ada, Anda mungkin perlu memperbarui add-on tertentu. Harap dicatat:

- Ini hanya berlaku untuk klaster yang ada yang beralih ke Mode Otomatis EKS.
- Cluster baru yang dibuat dengan Mode Otomatis EKS diaktifkan tidak memerlukan pembaruan ini.

Jika Anda memiliki salah satu add-on berikut yang diinstal, pastikan mereka setidaknya pada versi minimum yang ditentukan:

Nama add-on	Versi minimum yang diperlukan
Plugin Amazon VPC CNI untuk Kubernetes	v1.19.0-eksbuild.1
Proksi Kube	• v1.26.15-eksbuild.19 • v1.27.16-eksbuild.14 • v1.28.15-eksbuild.4 • v1.29.10-eksbuild.3

Nama add-on	Versi minimum yang diperlukan
	- v1.30.6-eksbuild.3 - v1.31.2-eksbuild.3
Driver CSI Amazon EBS	v1.37.0-eksbuild.1
Pengontrol snapshot CSI	v8.1.0-eksbuild.2
Agen Identitas EKS Pod	v1.3.4-eksbuild.1

Untuk informasi selengkapnya, lihat [the section called “Perbarui add-on”](#).

Langkah Berikutnya

- Untuk memigrasikan beban kerja Kelola Grup Node, lihat [the section called “Migrasi dari MNGs”](#)
- Untuk bermigrasi dari Karpenter yang Dikelola Sendiri, lihat [the section called “Migrasi dari Karpenter”](#)

Migrasi dari Karpenter ke Mode Otomatis EKS menggunakan kubectl

Topik ini memandu Anda melalui proses migrasi beban kerja dari Karpenter ke Amazon EKS Auto Mode menggunakan kubectl. Migrasi dapat dilakukan secara bertahap, memungkinkan Anda untuk memindahkan beban kerja dengan kecepatan Anda sendiri sambil menjaga stabilitas kluster dan ketersediaan aplikasi selama transisi.

step-by-step Pendekatan yang diuraikan di bawah ini memungkinkan Anda menjalankan Mode Otomatis Karpenter dan EKS secara berdampingan selama periode migrasi. Strategi operasi ganda ini membantu memastikan transisi yang mulus dengan memungkinkan Anda memvalidasi perilaku beban kerja pada Mode Otomatis EKS sebelum sepenuhnya menonaktifkan Karpenter. Anda dapat memigrasikan aplikasi secara individu atau dalam kelompok, memberikan fleksibilitas untuk mengakomodasi persyaratan operasional spesifik dan toleransi risiko Anda.

Prasyarat

Sebelum memulai migrasi, pastikan Anda memiliki:

- Karpenter v1.1 atau yang lebih baru diinstal pada cluster Anda. Untuk informasi selengkapnya, lihat [Memutakhirkan ke 1.1.0+](#) di dokumen Karpenter.

- `kubectl` diinstal dan terhubung ke cluster Anda. Untuk informasi selengkapnya, lihat [Penyiapan](#).

Topik ini mengasumsikan Anda akrab dengan Karpenter dan NodePools. Untuk informasi lebih lanjut, lihat Dokumentasi [Karpenter](#).

Langkah 1: Aktifkan Mode Otomatis EKS di cluster

Aktifkan Mode Otomatis EKS di cluster Anda yang ada menggunakan AWS CLI atau Management Console. Untuk informasi selengkapnya, lihat [the section called “Aktifkan di cluster”](#).

Note

Saat mengaktifkan Mode Otomatis EKS, jangan aktifkan `general purpose` nodepool pada tahap ini selama transisi. Kumpulan node ini tidak selektif.

Untuk informasi selengkapnya, lihat [the section called “Tinjau kumpulan node bawaan”](#).

Langkah 2: Buat Mode Otomatis EKS yang tercemar NodePool

Buat yang baru NodePool untuk Mode Otomatis EKS dengan `noda`. Ini memastikan bahwa pod yang ada tidak akan secara otomatis menjadwalkan pada node Mode Otomatis EKS yang baru. Kumpulan node ini menggunakan Mode Otomatis EKS default NodeClass bawaan. Untuk informasi selengkapnya, lihat [the section called “Buat kelas simpul”](#).

Contoh kumpulan node dengan taint:

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: eks-auto-mode
spec:
  template:
    spec:
      requirements:
        - key: "eks.amazonaws.com/instance-category"
          operator: In
          values: ["c", "m", "r"]
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
```

```
name: default
taints:
  - key: "eks-auto-mode"
    effect: "NoSchedule"
```

Perbarui persyaratan untuk kumpulan node agar sesuai dengan konfigurasi Karpenter tempat Anda bermigrasi. Anda membutuhkan setidaknya satu persyaratan.

Langkah 3: Perbarui beban kerja untuk migrasi

Identifikasi dan perbarui beban kerja yang ingin Anda migrasi ke Mode Otomatis EKS. Tambahkan toleransi dan pemilih node ke beban kerja ini:

```
apiVersion: apps/v1
kind: Deployment
spec:
  template:
    spec:
      tolerations:
        - key: "eks-auto-mode"
          effect: "NoSchedule"
      nodeSelector:
        eks.amazonaws.com/compute-type: auto
```

Perubahan ini memungkinkan beban kerja dijadwalkan pada node Mode Otomatis EKS yang baru.

Mode Otomatis EKS menggunakan label yang berbeda dari Karpenter. Label yang terkait dengan instance EC2 terkelola dimulai dengan `eks.amazonaws.com`. Untuk informasi selengkapnya, lihat [the section called “Buat kumpulan simpul”](#).

Langkah 4: Migrasikan beban kerja secara bertahap

Ulangi Langkah 3 untuk setiap beban kerja yang ingin Anda migrasikan. Ini memungkinkan Anda untuk memindahkan beban kerja secara individu atau dalam kelompok, berdasarkan persyaratan dan toleransi risiko Anda.

Langkah 5: Hapus Karpenter asli NodePool

Setelah semua beban kerja dimigrasikan, Anda dapat menghapus Karpenter asli: NodePool

```
kubectl delete nodepool <original-nodepool-name>
```

Langkah 6: Hapus noda dari Mode Otomatis EKS NodePool (Opsional)

Jika Anda ingin Mode Otomatis EKS menjadi default untuk beban kerja baru, Anda dapat menghapus noda dari Mode Otomatis EKS: NodePool

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: eks-auto-mode
spec:
  template:
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default
      # Remove the taints section
```

Langkah 7: Hapus pemilih node dari beban kerja (Opsional)

Jika Anda telah menghapus noda dari Mode Otomatis EKS NodePool, Anda dapat secara opsional menghapus pemilih node dari beban kerja Anda, karena Mode Otomatis EKS sekarang menjadi default:

```
apiVersion: apps/v1
kind: Deployment
spec:
  template:
    spec:
      # Remove the nodeSelector section
      tolerations:
        - key: "eks-auto-mode"
          effect: "NoSchedule"
```

Langkah 8: Uninstall Karpenter dari cluster Anda

Langkah-langkah untuk menghapus Karpenter tergantung pada bagaimana Anda menginstalnya. Untuk informasi lebih lanjut, lihat petunjuk [pemasangan Karpenter](#).

Migrasi dari Grup Node Terkelola EKS ke Mode Otomatis EKS

Saat mentransisikan kluster Amazon EKS untuk menggunakan mode otomatis EKS, Anda dapat dengan lancar memigrasikan beban kerja yang ada dari grup node terkelola (MNGs) menggunakan alat CLI eksctl. Proses ini memastikan ketersediaan aplikasi berkelanjutan sementara mode otomatis EKS mengoptimalkan sumber daya komputasi Anda. Migrasi dapat dilakukan dengan gangguan minimal pada aplikasi yang sedang berjalan.

Topik ini memandu Anda melalui langkah-langkah untuk menguras pod dengan aman dari grup node terkelola yang ada dan memungkinkan mode otomatis EKS untuk menjadwalkannya kembali pada instance yang baru disediakan. Dengan mengikuti prosedur ini, Anda dapat memanfaatkan konsolidasi beban kerja cerdas mode otomatis EKS sambil mempertahankan ketersediaan aplikasi Anda selama migrasi.

Prasyarat

- Cluster dengan Mode Otomatis EKS diaktifkan
- eksctl CLI diinstal dan terhubung ke cluster Anda. Untuk informasi selengkapnya, lihat [Penyiapan](#).
- Karpenter tidak diinstal pada cluster.

Prosedur

Gunakan perintah eksctl CLI berikut untuk memulai pengeringan pod dari instance grup node terkelola yang ada. Mode Otomatis EKS akan membuat node baru untuk mendukung pod yang dipindahkan.

```
eksctl delete nodegroup --cluster=<clusterName> --name=<nodegroupName>
```

Anda harus menjalankan perintah ini untuk setiap grup node terkelola di cluster Anda.

Untuk informasi selengkapnya tentang perintah ini, lihat [Menghapus dan menguras nodegroup](#) di dokumen eksctl.

Bermigrasi dari EKS Fargate ke Mode Otomatis EKS

Topik ini memandu Anda melalui proses migrasi beban kerja dari EKS Fargate ke Amazon EKS Auto Mode menggunakan kubectl. Migrasi dapat dilakukan secara bertahap, memungkinkan Anda untuk

memindahkan beban kerja dengan kecepatan Anda sendiri sambil menjaga stabilitas kluster dan ketersediaan aplikasi selama transisi.

step-by-step Pendekatan yang diuraikan di bawah ini memungkinkan Anda menjalankan EKS Fargate dan Mode Otomatis EKS berdampingan selama periode migrasi. Strategi operasi ganda ini membantu memastikan transisi yang mulus dengan memungkinkan Anda memvalidasi perilaku beban kerja pada Mode Otomatis EKS sebelum sepenuhnya menonaktifkan EKS Fargate. Anda dapat memigrasikan aplikasi secara individu atau dalam kelompok, memberikan fleksibilitas untuk mengakomodasi persyaratan operasional spesifik dan toleransi risiko Anda.

Membandingkan Amazon EKS Auto Mode dan EKS dengan AWS Fargate

Amazon EKS dengan AWS Fargate tetap menjadi pilihan bagi pelanggan yang ingin menjalankan EKS, tetapi Amazon EKS Auto Mode adalah pendekatan yang disarankan untuk bergerak maju. Mode Otomatis EKS sepenuhnya sesuai dengan Kubernetes, mendukung semua primitif Kubernetes hulu dan alat platform seperti Istio, yang tidak dapat didukung Fargate. Mode Otomatis EKS juga sepenuhnya mendukung semua opsi pembelian EC2 runtime, termasuk instans GPU dan Spot, memungkinkan pelanggan untuk memanfaatkan EC2 diskon yang dinegosiasikan dan mekanisme penghematan lainnya. Kemampuan ini tidak tersedia saat menggunakan EKS dengan Fargate.

Selain itu, Mode Otomatis EKS memungkinkan pelanggan untuk mencapai model isolasi yang sama dengan Fargate, menggunakan kemampuan penjadwalan Kubernetes standar untuk memastikan setiap EC2 instance menjalankan satu wadah aplikasi. Dengan mengadopsi Amazon EKS Auto Mode, pelanggan dapat membuka manfaat penuh dari menjalankan Kubernetes AWS — platform Kubernetes-Conformant sepenuhnya yang memberikan fleksibilitas untuk memanfaatkan seluruh luasnya dan opsi pembelian sambil mempertahankan kemudahan penggunaan EC2 dan abstraksi dari manajemen infrastruktur yang disediakan Fargate.

Prasyarat

Sebelum memulai migrasi, pastikan Anda memiliki

- Siapkan cluster dengan Fargate. Untuk informasi selengkapnya, lihat [the section called “Memulai”](#).
- Diinstal dan terhubung kubectl ke cluster Anda. Untuk informasi selengkapnya, lihat [Penyiapan](#).

Langkah 1: Periksa cluster Fargate

1. Periksa apakah cluster EKS dengan Fargate sedang berjalan:

```
kubectl get node
```

```
NAME STATUS ROLES AGE VERSION
fargate-ip-192-168-92-52.ec2.internal Ready <none> 25m v1.30.8-eks-2d5f260
fargate-ip-192-168-98-196.ec2.internal Ready <none> 24m v1.30.8-eks-2d5f260
```

2. Periksa pod yang sedang berjalan:

```
kubectl get pod -A
```

```
NAMESPACE NAME READY STATUS RESTARTS AGE
kube-system coredns-6659cb98f6-gxpjz 1/1 Running 0 26m
kube-system coredns-6659cb98f6-gzzsx 1/1 Running 0 26m
```

3. Buat penyebaran dalam file bernama deployment_fargate.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    annotations:
      eks.amazonaws.com/compute-type: fargate
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
```

4. Terapkan penyebaran:


```
kubectl apply -f deployment_fargate.yaml
```

```
deployment.apps/nginx-deployment created
```

5. Periksa pod dan penerapan:

```
kubectl get pod,deploy
```

NAME	READY	STATUS	RESTARTS	AGE
pod/nginx-deployment-5c7479459b-6trtm	1/1	Running	0	61s
pod/nginx-deployment-5c7479459b-g8ssb	1/1	Running	0	61s
pod/nginx-deployment-5c7479459b-mq4mf	1/1	Running	0	61s

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/nginx-deployment	3/3	3	3	61s

6. Periksa simpul:

```
kubectl get node -owide
```

NAME	INTERNAL-IP	EXTERNAL-IP	OS-IMAGE	STATUS	ROLES	AGE	VERSION
fargate-ip-192-168-111-43	192.168.111.43	<none>	Amazon Linux 2	Ready	<none>	31s	v1.30.8-eks-2d5f260
containerd://1.7.25							
fargate-ip-192-168-117-130	192.168.117.130	<none>	Amazon Linux 2	Ready	<none>	36s	v1.30.8-eks-2d5f260
containerd://1.7.25							
fargate-ip-192-168-74-140	192.168.74.140	<none>	Amazon Linux 2	Ready	<none>	36s	v1.30.8-eks-2d5f260
containerd://1.7.25							

Langkah 2: Aktifkan Mode Otomatis EKS di cluster

1. Aktifkan Mode Otomatis EKS di cluster Anda yang ada menggunakan AWS CLI atau Management Console. Untuk informasi selengkapnya, lihat [the section called “Aktifkan di cluster”](#).
2. Periksa nodepool:

```
kubectl get nodepool
```

NAME	NODECLASS	NODES	READY	AGE
general-purpose	default	1	True	6m58s
system	default	0	True	3d14h

Langkah 3: Perbarui beban kerja untuk migrasi

Identifikasi dan perbarui beban kerja yang ingin Anda migrasi ke Mode Otomatis EKS.

Untuk memigrasikan beban kerja dari Fargate ke Mode Otomatis EKS, terapkan anotasi.

`eks.amazonaws.com/compute-type: ec2` Ini memastikan bahwa beban kerja tidak akan dijadwalkan oleh Fargate, terlepas dari profil Fargate, dan akan terjebak oleh Mode Otomatis EKS.

NodePool Untuk informasi selengkapnya, lihat [the section called “Buat kumpulan simpul”](#).

1. Ubah penerapan Anda (misalnya, `deployment_fargate.yaml` file) untuk mengubah jenis komputasi menjadi: `ec2`

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
      annotations:
        eks.amazonaws.com/compute-type: ec2
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
```

```
- containerPort: 80
```

2. Terapkan penyebaran. Perubahan ini memungkinkan beban kerja dijadwalkan pada node Mode Otomatis EKS yang baru:

```
kubectl apply -f deployment_fargate.yaml
```

3. Periksa apakah penerapan berjalan di kluster Mode Otomatis EKS:

```
kubectl get pod -o wide
```

NAME	READY	STATUS	RESTARTS	AGE	IP
NODE	NOMINATED	NODE	READINESS	GATES	
nginx-deployment-97967b68d-ffxxh	1/1	Running	0	3m31s	
192.168.43.240	i-0845aafcb51630ffb	<none>	<none>		
nginx-deployment-97967b68d-mbcgj	1/1	Running	0	2m37s	
192.168.43.241	i-0845aafcb51630ffb	<none>	<none>		
nginx-deployment-97967b68d-qpd8x	1/1	Running	0	2m35s	
192.168.43.242	i-0845aafcb51630ffb	<none>	<none>		

4. Pastikan tidak ada node Fargate yang berjalan dan penerapan yang berjalan di node terkelola Mode Otomatis EKS:

```
kubectl get node -owide
```

NAME	STATUS	ROLES	AGE	VERSION	INTERNAL-IP	EXTERNAL-IP
OS-IMAGE				KERNEL-VERSION	CONTAINER-RUNTIME	
i-0845aafcb51630ffb	Ready	<none>	3m30s	v1.30.8-eks-3c20087	192.168.41.125	
3.81.118.95	Bottlerocket	(EKS Auto)	2025.3.14	(aws-k8s-1.30)	6.1.129	
containerd://1.7.25+bottlerocket						

Langkah 4: Migrasikan beban kerja secara bertahap

Ulangi Langkah 3 untuk setiap beban kerja yang ingin Anda migrasikan. Ini memungkinkan Anda untuk memindahkan beban kerja secara individu atau dalam kelompok, berdasarkan persyaratan dan toleransi risiko Anda.

Langkah 5: Hapus profil fargate asli

Setelah semua beban kerja dimigrasikan, Anda dapat menghapus profil aslinya `fargate`. Ganti `<fargate profile name>` dengan nama profil Fargate Anda:

```
aws eks delete-fargate-profile --cluster-name eks-fargate-demo-cluster --fargate-profile-name <fargate profile name>
```

Langkah 6: Turunkan CoreDNS

Karena mode EKS Auto menangani CoreDNS, Anda menskalakan penerapan `coredns` ke 0:

```
kubectl scale deployment coredns -n kube-system --replicas=0
```

Jalankan beban kerja sampel di kluster Mode Otomatis EKS

Bab ini memberikan contoh cara menerapkan berbagai jenis beban kerja ke kluster Amazon EKS yang berjalan dalam Mode Otomatis. Contoh menunjukkan pola beban kerja utama termasuk aplikasi sampel, aplikasi web yang seimbang beban, beban kerja stateful menggunakan penyimpanan persisten, dan beban kerja dengan persyaratan penempatan node tertentu. Setiap contoh mencakup manifes lengkap dan instruksi step-by-step penyebaran yang dapat Anda gunakan sebagai templat untuk aplikasi Anda sendiri.

Sebelum melanjutkan dengan contoh, pastikan bahwa Anda memiliki kluster EKS yang berjalan dalam Mode Otomatis dan bahwa Anda telah menginstal CLI dan AWS `kubectl`. Untuk informasi selengkapnya, lihat [Penyiapan](#). Contohnya mengasumsikan keakraban dasar dengan konsep Kubernetes dan perintah `kubectl`.

Anda dapat menggunakan sampel berbasis kasus penggunaan ini untuk menjalankan beban kerja di kluster Mode Otomatis EKS.

[the section called “Menyebarkan beban kerja yang mengembang”](#)

Menunjukkan cara menerapkan beban kerja sampel ke kluster Mode Otomatis EKS menggunakan `kubectl` perintah.

[the section called “Menyebarkan penyeimbang beban”](#)

Menunjukkan cara menerapkan versi kontainer dari game 2048 di Amazon EKS.

[the section called “Menyebarkan beban kerja stateful”](#)

Menunjukkan cara menerapkan contoh aplikasi stateful ke kluster Mode Otomatis EKS.

[the section called “Kontrol penyebaran”](#)

Menunjukkan cara menggunakan anotasi untuk mengontrol jika beban kerja diterapkan ke node yang dikelola oleh Mode Otomatis EKS.

Menerapkan sampel beban kerja inflate ke kluster Mode Otomatis Amazon EKS

Dalam tutorial ini, Anda akan mempelajari cara menerapkan beban kerja sampel ke cluster Mode Otomatis EKS dan mengamati bagaimana ia secara otomatis menyediakan sumber daya komputasi yang diperlukan. Anda akan menggunakan `kubectl` perintah untuk melihat perilaku kluster dan melihat secara langsung bagaimana Mode Otomatis menyederhanakan operasi Kubernetes. AWS Pada akhir tutorial ini, Anda akan memahami bagaimana EKS Auto Mode merespons penerapan beban kerja dengan secara otomatis mengelola sumber daya komputasi yang mendasarinya, tanpa memerlukan konfigurasi grup node manual.

Prasyarat

- Cluster Mode Otomatis Amazon EKS. Perhatikan nama dan AWS wilayah cluster.
- Prinsipal IAM, seperti pengguna atau peran, dengan izin yang cukup untuk mengelola jaringan, komputasi, dan sumber daya EKS.
 - Untuk informasi selengkapnya, lihat [Membuat peran dan melampirkan kebijakan di Panduan Pengguna IAM di Panduan Pengguna IAM](#).
- `awsCLI` diinstal dan dikonfigurasi dengan identitas IAM.
- `kubectl` CLI diinstal dan terhubung ke cluster.
 - Untuk informasi selengkapnya, lihat [Penyiapan](#).

Langkah 1: Tinjau sumber daya komputasi yang ada (opsional)

Pertama, gunakan `kubectl` untuk membuat daftar kumpulan node di cluster Anda.

```
kubectl get nodepools
```

Keluaran sampel:

```
general-purpose
```

Dalam tutorial ini, kita akan menyebarkan beban kerja yang dikonfigurasi untuk menggunakan `general-purpose` node pool. Kumpulan node ini dibangun ke dalam Mode Otomatis EKS, dan mencakup default yang wajar untuk beban kerja umum, seperti layanan mikro dan aplikasi web. Anda dapat membuat kumpulan node Anda sendiri. Untuk informasi selengkapnya, lihat [the section called “Buat kumpulan simpul”](#).

Kedua, gunakan `kubectl` untuk membuat daftar node yang terhubung ke cluster Anda.

```
kubectl get nodes
```

Jika Anda baru saja membuat kluster Mode Otomatis EKS, Anda tidak akan memiliki node.

Dalam tutorial ini Anda akan menyebarkan beban kerja sampel. Jika Anda tidak memiliki node, atau beban kerja tidak dapat muat pada node yang ada, Mode Otomatis EKS akan menyediakan node baru.

Langkah 2: Menyebarakan aplikasi sampel ke cluster

Tinjau Deployment Kubernetes berikut dan simpan sebagai `inflate.yaml`

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: inflate
spec:
  replicas: 1
  selector:
    matchLabels:
      app: inflate
  template:
    metadata:
      labels:
        app: inflate
    spec:
      terminationGracePeriodSeconds: 0
      nodeSelector:
        eks.amazonaws.com/compute-type: auto
      securityContext:
        runAsUser: 1000
```

```
runAsGroup: 3000
fsGroup: 2000
containers:
- name: inflate
  image: public.ecr.aws/eks-distro/kubernetes/pause:3.7
  resources:
    requests:
      cpu: 1
  securityContext:
    allowPrivilegeEscalation: false
```

Perhatikan bahwa `eks.amazonaws.com/compute-type: auto` memilih mengharuskan beban kerja diterapkan pada node Mode Otomatis Amazon EKS.

Terapkan Deployment ke cluster Anda.

```
kubectl apply -f inflate.yaml
```

Langkah 3: Tonton Acara Kubernetes

Gunakan perintah berikut untuk menonton peristiwa Kubernetes, termasuk membuat node baru. Gunakan `ctrl+c` untuk berhenti menonton acara.

```
kubectl get events -w --sort-by '.lastTimestamp'
```

Gunakan `kubectl` untuk membuat daftar node yang terhubung ke cluster Anda lagi. Perhatikan node yang baru dibuat.

```
kubectl get nodes
```

Langkah 4: Lihat node dan instance di konsol AWS

Anda dapat melihat Node Mode Otomatis EKS di konsol EKS, dan EC2 instance terkait di EC2 konsol.

EC2 Instans yang digunakan oleh Mode Otomatis EKS dibatasi. Anda tidak dapat menjalankan perintah arbitrer pada node Mode Otomatis EKS.

Langkah 5: Hapus penyebaran

Gunakan `kubectl` untuk menghapus penyebaran sampel

```
kubectl delete -f inflate.yaml
```

Jika Anda tidak memiliki beban kerja lain yang diterapkan ke cluster Anda, node yang dibuat oleh Mode Otomatis EKS akan kosong.

Dalam konfigurasi default, Mode Otomatis EKS mendeteksi node yang telah kosong selama tiga puluh detik, dan menghentikannya.

Gunakan `kubectl` atau EC2 konsol untuk mengonfirmasi instance terkait telah dihapus.

Menerapkan Beban Kerja Load Balancer Sampel ke Mode Otomatis EKS

Panduan ini memandu Anda melalui penerapan versi kontainer dari game 2048 di Amazon EKS, lengkap dengan penyeimbangan beban dan aksesibilitas internet.

Prasyarat

- Kluster Mode Otomatis EKS
- `kubectl` dikonfigurasi untuk berinteraksi dengan cluster Anda
- Izin IAM yang sesuai untuk membuat sumber daya ALB

Langkah 1: Buat Namespace

Pertama, buat namespace khusus untuk aplikasi game 2048.

Buat file bernama `01-namespace.yaml`:

```
apiVersion: v1
kind: Namespace
metadata:
  name: game-2048
```

Terapkan konfigurasi namespace:

```
kubectl apply -f 01-namespace.yaml
```

Langkah 2: Menyebarkan Aplikasi

Aplikasi ini menjalankan beberapa replika dari kontainer game 2048.

Buat file bernama `02-deployment.yaml`:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: game-2048
  name: deployment-2048
spec:
  selector:
    matchLabels:
      app.kubernetes.io/name: app-2048
  replicas: 5
  template:
    metadata:
      labels:
        app.kubernetes.io/name: app-2048
    spec:
      containers:
        - image: public.ecr.aws/16m2t8p7/docker-2048:latest
          imagePullPolicy: Always
          name: app-2048
          ports:
            - containerPort: 80
      resources:
        requests:
          cpu: "0.5"
```

Note

Jika Anda menerima kesalahan saat memuat gambar `public.ecr.aws/16m2t8p7/docker-2048:latest`, konfirmasi peran IAM Node Anda memiliki izin yang cukup untuk menarik gambar dari ECR. Untuk informasi selengkapnya, lihat [the section called “IAM role simpul”](#). Juga, `docker-2048` gambar dalam contoh adalah x86_64 gambar dan tidak akan berjalan pada arsitektur lain.

Komponen kunci:

- Menyebarkan 5 replika aplikasi
- Menggunakan gambar ECR publik
- Meminta 0,5 inti CPU per pod

- Mengekspos port 80 untuk lalu lintas HTTP

Terapkan penyebaran:

```
kubectl apply -f 02-deployment.yaml
```

Langkah 3: Buat Layanan

Layanan mengekspos penyebaran ke jaringan cluster.

Buat file bernama `03-service.yaml`:

```
apiVersion: v1
kind: Service
metadata:
  namespace: game-2048
  name: service-2048
spec:
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
  selector:
    app.kubernetes.io/name: app-2048
```

Komponen kunci:

- Menciptakan NodePort layanan
- Memetakan port 80 ke port kontainer 80
- Menggunakan pemilih label untuk menemukan pod

Terapkan layanan:

```
kubectl apply -f 03-service.yaml
```

Langkah 4: Konfigurasi Load Balancing

Anda akan mengatur ingress untuk mengekspos aplikasi ke internet.

Pertama, buat `IngressClass`. Buat file bernama `04-ingressclass.yaml`:

```
apiVersion: networking.k8s.io/v1
kind: IngressClass
metadata:
  labels:
    app.kubernetes.io/name: LoadBalancerController
  name: alb
spec:
  controller: eks.amazonaws.com/alb
```

 Note

Mode Otomatis EKS memerlukan tag subnet untuk mengidentifikasi subnet publik dan pribadi.

Jika Anda membuat cluster Anda dengan `eksctl`, Anda sudah memiliki tag ini.

Pelajari cara [the section called “Tag subnet”](#).

Kemudian buat sumber daya `Ingress`. Buat file bernama `05-ingress.yaml`:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  namespace: game-2048
  name: ingress-2048
  annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
spec:
  ingressClassName: alb
  rules:
    - http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: service-2048
                port:
                  number: 80
```

Komponen kunci:

- Menciptakan ALB yang menghadap ke internet
- Menggunakan tipe target IP untuk routing pod langsung
- Rutekan semua lalu lintas (/) ke layanan game

Terapkan konfigurasi ingress:

```
kubectl apply -f 04-ingressclass.yaml
kubectl apply -f 05-ingress.yaml
```

Langkah 5: Verifikasi Deployment

1. Periksa apakah semua pod berjalan:

```
kubectl get pods -n game-2048
```

2. Verifikasi layanan dibuat:

```
kubectl get svc -n game-2048
```

3. Dapatkan titik akhir ALB:

```
kubectl get ingress -n game-2048
```

Bidang ADDRESS pada keluaran ingress akan menampilkan endpoint ALB Anda. Tunggu 2-3 menit hingga ALB menyediakan dan mendaftarkan semua target.

Langkah 6: Akses Game

Buka browser web Anda dan telusuri URL endpoint ALB dari langkah sebelumnya. Anda akan melihat antarmuka game 2048.

Langkah 7: Pembersihan

Untuk menghapus semua sumber daya yang dibuat dalam tutorial ini:

```
kubectl delete namespace game-2048
```

Ini akan menghapus semua sumber daya di namespace, termasuk penerapan, layanan, dan sumber daya ingress.

Apa yang Terjadi Di Balik Layar

1. Deployment menciptakan 5 pod yang menjalankan game 2048
2. Layanan ini menyediakan akses jaringan yang stabil ke pod ini
3. Mode Otomatis EKS:
 - Membuat Application Load Balancer di AWS
 - Mengonfigurasi grup target untuk pod
 - Menyiapkan aturan perutean untuk mengarahkan lalu lintas ke layanan

Pemecahan Masalah

Jika game tidak dimuat:

- Pastikan semua pod berjalan: `kubectl get pods -n game-2048`
- Periksa status masuknya: `kubectl describe ingress -n game-2048`
- Verifikasi pemeriksaan kesehatan ALB: Periksa kesehatan grup target di Konsol AWS

Menerapkan contoh beban kerja stateful ke Mode Otomatis EKS

Tutorial ini akan memandu Anda melalui penerapan contoh aplikasi stateful ke kluster Mode Otomatis EKS Anda. Aplikasi ini menulis stempel waktu ke volume persisten, mendemonstrasikan penyediaan volume EBS otomatis dan kemampuan persistensi EKS Auto Mode.

Prasyarat

- Kluster Mode Otomatis EKS
- AWS CLI dikonfigurasi dengan izin yang sesuai
- `kubectl` diinstal dan dikonfigurasi
 - Untuk informasi selengkapnya, lihat [Penyiapan](#).

Langkah 1: Konfigurasi lingkungan Anda

1. Tetapkan variabel lingkungan Anda:

```
export CLUSTER_NAME=my-auto-cluster
export AWS_REGION="us-west-2"
```

2. Perbarui kubeconfig Anda:

```
aws eks update-kubeconfig --name "${CLUSTER_NAME}"
```

Langkah 2: Buat kelas penyimpanan

Ini StorageClass menentukan bagaimana Mode Otomatis EKS akan menyediakan volume EBS.

Mode Otomatis EKS tidak membuat StorageClass untuk Anda. Anda harus membuat StorageClass referensi `ebs.csi.eks.amazonaws.com` untuk menggunakan kemampuan penyimpanan Mode Otomatis EKS.

1. Buat file bernama `storage-class.yaml`:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: auto-ebs-sc
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
provisioner: ebs.csi.eks.amazonaws.com
volumeBindingMode: WaitForFirstConsumer
parameters:
  type: gp3
  encrypted: "true"
```

2. Terapkan StorageClass:

```
kubectl apply -f storage-class.yaml
```

Komponen kunci:

- `provisioner: ebs.csi.eks.amazonaws.com`- Menggunakan Mode Otomatis EKS
- `volumeBindingMode: WaitForFirstConsumer`- Menunda pembuatan volume hingga pod membutuhkannya

- `type: gp3`- Menentukan jenis volume EBS
- `encrypted: "true"`- EBS akan menggunakan `aws/ebs` kunci default untuk mengenkripsi volume yang dibuat dengan kelas ini. Ini memang opsional, tetapi direkomendasikan.
- `storageclass.kubernetes.io/is-default-class: "true"`- Kubernetes akan menggunakan kelas penyimpanan ini secara default, kecuali Anda menentukan kelas volume yang berbeda pada klaim volume persisten. Berhati-hatilah saat menyetel nilai ini jika Anda bermigrasi dari pengontrol penyimpanan lain. (opsional)

Langkah 3: Buat klaim volume persisten

PVC meminta penyimpanan dari `StorageClass`.

1. Buat file bernama `pvc.yaml`:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: auto-ebs-claim
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: auto-ebs-sc
  resources:
    requests:
      storage: 8Gi
```

2. Terapkan PVC:

```
kubectl apply -f pvc.yaml
```

Komponen kunci:

- `accessModes: ReadWriteOnce`- Volume dapat dipasang oleh satu node pada satu waktu
- `storage: 8Gi`- Meminta volume 8 GiB
- `storageClassName: auto-ebs-sc`- Referensi yang `StorageClass` kami buat

Langkah 4: Menyebarakan Aplikasi

Deployment menjalankan container yang menulis stempel waktu ke volume persisten.

1. Buat file bernama `deployment.yaml`:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: inflate-stateful
spec:
  replicas: 1
  selector:
    matchLabels:
      app: inflate-stateful
  template:
    metadata:
      labels:
        app: inflate-stateful
    spec:
      terminationGracePeriodSeconds: 0
      nodeSelector:
        eks.amazonaws.com/compute-type: auto
      containers:
        - name: bash
          image: public.ecr.aws/docker/library/bash:4.4
          command: ["/usr/local/bin/bash"]
          args: ["-c", "while true; do echo $(date -u) >> /data/out.txt; sleep 60;
done"]
      resources:
        requests:
          cpu: "1"
        volumeMounts:
          - name: persistent-storage
            mountPath: /data
      volumes:
        - name: persistent-storage
          persistentVolumeClaim:
            claimName: auto-ebs-claim
```

2. Terapkan Deployment:

```
kubectl apply -f deployment.yaml
```


Komponen kunci:

- Kontainer bash sederhana yang menulis stempel waktu ke file
- Memasang PVC di /data
- Permintaan 1 inti CPU
- Menggunakan pemilih node untuk node terkelola EKS

Langkah 5: Verifikasi Pengaturan

1. Periksa apakah pod sedang berjalan:

```
kubectl get pods -l app=inflate-stateful
```

2. Verifikasi PVC terikat:

```
kubectl get pvc auto-ebs-claim
```

3. Periksa volume EBS:

```
# Get the PV name
PV_NAME=$(kubectl get pvc auto-ebs-claim -o jsonpath='{.spec.volumeName}')
# Describe the EBS volume
aws ec2 describe-volumes \
  --filters Name=tag:CSIVolumeName,Values=${PV_NAME}
```

4. Verifikasi data sedang ditulis:

```
kubectl exec "$(kubectl get pods -l app=inflate-stateful \
  -o=jsonpath='{.items[0].metadata.name}')" -- \
  cat /data/out.txt
```

Langkah 6: Pembersihan

Jalankan perintah berikut untuk menghapus semua sumber daya yang dibuat dalam tutorial ini:

```
# Delete all resources in one command
kubectl delete deployment/inflate-stateful pvc/auto-ebs-claim storageclass/auto-ebs-sc
```

Apa yang Terjadi Di Balik Layar

1. PVC meminta penyimpanan dari StorageClass
2. Saat Pod dijadwalkan:
 - a. Mode Otomatis EKS menyediakan volume EBS
 - b. Menciptakan PersistentVolume
 - c. Menempelkan volume ke node
3. Pod memasang volume dan mulai menulis stempel waktu

Pengontrol Snapshot

EKS Auto Mode kompatibel dengan Kubernetes CSI Snapshotter, juga dikenal sebagai pengontrol snapshot. Namun, Mode Otomatis EKS tidak menyertakan pengontrol snapshot. Anda bertanggung jawab untuk menginstal dan mengkonfigurasi pengontrol snapshot. Untuk informasi selengkapnya, lihat [the section called “Pengontrol snapshot CSI”](#).

Tinjau hal-hal berikut VolumeSnapshotClass yang mereferensikan kemampuan penyimpanan Mode Otomatis EKS.

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: auto-ebs-vsclass
driver: ebs.csi.eks.amazonaws.com
deletionPolicy: Delete
```

[Pelajari lebih lanjut tentang Snapshotter CSI Kubernetes.](#)

Konfigurasi pengaturan Mode Otomatis EKS

Bab ini menjelaskan cara mengonfigurasi aspek-aspek tertentu dari kluster Mode Otomatis Amazon Elastic Kubernetes Service (EKS) Anda. Sementara Mode Otomatis EKS mengelola sebagian besar komponen infrastruktur secara otomatis, Anda dapat menyesuaikan fitur tertentu untuk memenuhi persyaratan beban kerja Anda.

Dengan menggunakan opsi konfigurasi yang dijelaskan dalam topik ini, Anda dapat memodifikasi pengaturan jaringan, menghitung sumber daya, dan perilaku penyeimbangan beban sambil mempertahankan manfaat manajemen infrastruktur otomatis. Sebelum membuat perubahan

konfigurasi, tinjau opsi yang tersedia di bagian berikut untuk menentukan pendekatan mana yang paling sesuai dengan kebutuhan Anda.

Fitur apa yang ingin Anda konfigurasi?	Opsi Konfigurasi
Jaringan dan penyimpanan simpul • Konfigurasi penempatan node di seluruh subnet publik dan pribadi • Tentukan grup keamanan khusus untuk kontrol akses node • Sesuaikan kebijakan terjemahan alamat jaringan (SNAT) • Aktifkan kebijakan jaringan Kubernetes, pencatatan dan pemantauan kebijakan jaringan yang terperinci • Tetapkan parameter penyimpanan sementara (ukuran, IOPS, throughput) • Konfigurasi penyimpanan singkat terenkripsi dengan kunci KMS khusus • Mengisolasi lalu lintas pod dalam subnet terpisah dari node	the section called “Buat kelas simpul”
Sumber daya komputasi node • Pilih jenis dan keluarga EC2 instans tertentu • Tentukan arsitektur CPU (x86_64,) ARM64 • Konfigurasi jenis kapasitas (On-Demand, Spot) • Tentukan Availability Zone • Konfigurasi noda dan label node • Tetapkan jumlah node minimum dan maksimum	the section called “Buat kumpulan simpul”
Pengaturan Application Load Balancer	the section called “Buat kelas ingress”

Fitur apa yang ingin Anda konfigurasi?	Opsi Konfigurasi
• Menyebarkan penyeimbang beban internal atau yang menghadap ke internet • Konfigurasi penyeimbangan beban lintas zona • Tetapkan periode batas waktu idle • Aktifkan HTTP/2 dan dukungan WebSocket • Konfigurasi parameter pemeriksaan kesehatan • Tentukan pengaturan sertifikat TLS • Tentukan atribut grup target • Mengatur jenis alamat IP (IPv4, dual-stack)	
Pengaturan Network Load Balancer • Konfigurasi perutean IP pod langsung • Aktifkan penyeimbangan beban lintas zona • Setel batas waktu idle koneksi • Konfigurasi parameter pemeriksaan kesehatan • Tentukan penempatan subnet • Mengatur jenis alamat IP (IPv4, dual-stack) • Konfigurasi mempertahankan IP sumber klien • Tentukan atribut grup target	the section called “Buat layanan”

Fitur apa yang ingin Anda konfigurasi?	Opsi Konfigurasi
Pengaturan Kelas Penyimpanan • Tentukan jenis volume EBS (gp3, io1, io2, dll.) • Konfigurasi enkripsi volume dan penggunaan kunci KMS • Tetapkan IOPS dan parameter throughput • Tetapkan sebagai kelas penyimpanan default • Tentukan tag kustom untuk volume yang disediakan	the section called “Buat StorageClass”
Kontrol Penggunaan ODCR • Konfigurasi penerapan beban kerja ke Reservasi Kapasitas Sesuai EC2 Permintaan • Secara eksplisit memilih spesifik ODCRs berdasarkan ID untuk penggunaan kapasitas yang ditargetkan • Gunakan pilihan berbasis tag untuk menargetkan grup ODCRs • Filter ODCRs dengan memiliki AWS akun untuk skenario lintas akun • Kontrol apakah beban kerja secara otomatis menggunakan open ODCRs	the section called “Kontrol Penerapan ODCR”

Buat Kelas Node untuk Amazon EKS

Amazon EKS Node Classes adalah template yang menawarkan kontrol granular atas konfigurasi node terkelola Mode Otomatis EKS Anda. Kelas Node mendefinisikan pengaturan tingkat infrastruktur yang berlaku untuk grup node di kluster EKS Anda, termasuk konfigurasi jaringan, pengaturan penyimpanan, dan penandaan sumber daya. Topik ini menjelaskan cara membuat dan mengkonfigurasi Kelas Node untuk memenuhi persyaratan operasional spesifik Anda.

Saat Anda perlu menyesuaikan cara Mode Otomatis EKS menyediakan dan mengonfigurasi EC2 instance di luar pengaturan default, membuat Kelas Node memberi Anda kontrol yang tepat atas parameter infrastruktur penting. Misalnya, Anda dapat menentukan penempatan subnet pribadi untuk keamanan yang ditingkatkan, mengonfigurasi penyimpanan sementara instance untuk beban kerja yang sensitif terhadap kinerja, atau menerapkan penandaan khusus untuk alokasi biaya.

Buat Kelas Node

Untuk membuat `NodeClass`, ikuti langkah-langkah ini:

1. Buat file YAMM (misalnya, `nodeclass.yaml`) dengan konfigurasi Kelas Node Anda
2. Terapkan konfigurasi ke cluster Anda menggunakan `kubectl`
3. Referensi Kelas Node dalam konfigurasi Node Pool Anda. Untuk informasi selengkapnya, lihat [the section called “Buat kumpulan simpul”](#).

Anda perlu `kubectl` diinstal dan dikonfigurasi. Untuk informasi selengkapnya, lihat [Penyiapan](#).

Contoh Kelas Node Dasar

Berikut adalah contoh Kelas Node:

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: private-compute
spec:
  subnetSelectorTerms:
    - tags:
        Name: "private-subnet"
        kubernetes.io/role/internal-elb: "1"
  securityGroupSelectorTerms:
    - tags:
        Name: "eks-cluster-sg"
  ephemeralStorage:
    size: "160Gi"
```

Ini `NodeClass` meningkatkan jumlah penyimpanan fana pada node.

Terapkan konfigurasi ini dengan menggunakan:

```
kubectl apply -f nodeclass.yaml
```

Selanjutnya, referensi Kelas Node dalam konfigurasi Node Pool Anda. Untuk informasi selengkapnya, lihat [the section called “Buat kumpulan simpul”](#).

Buat entri akses kelas node

Jika Anda membuat kelas node kustom, Anda perlu membuat Entri Akses EKS untuk mengizinkan node bergabung dengan cluster. EKS secara otomatis membuat entri akses saat Anda menggunakan kelas node dan kumpulan node bawaan.

Untuk informasi tentang cara kerja Entri Akses, lihat [the section called “Berikan izin”](#).

Saat membuat entri akses untuk kelas node Mode Otomatis EKS, Anda perlu menggunakan jenis entri EC2 akses.

Buat entri akses dengan CLI

Untuk membuat entri akses untuk EC2 node dan mengaitkan Kebijakan Node Otomatis EKS:

Perbarui perintah CLI berikut dengan nama cluster Anda, dan peran node ARN. Peran node ARN ditentukan dalam kelas node YAMM.

```
# Create the access entry for EC2 nodes
aws eks create-access-entry \
  --cluster-name <cluster-name> \
  --principal-arn <node-role-arn> \
  --type EC2

# Associate the auto node policy
aws eks associate-access-policy \
  --cluster-name <cluster-name> \
  --principal-arn <node-role-arn> \
  --policy-arn arn:aws:eks::aws:cluster-access-policy/AmazonEKSAutoNodePolicy \
  --access-scope type=cluster
```

Buat entri akses dengan CloudFormation

Untuk membuat entri akses untuk EC2 node dan mengaitkan Kebijakan Node Otomatis EKS:

Perbarui yang berikut ini CloudFormation dengan nama cluster Anda, dan peran node ARN. Peran node ARN ditentukan dalam kelas node YAMM.

```

EKSAutoNodeRoleAccessEntry:
  Type: AWS::EKS::AccessEntry
  Properties:
    ClusterName: <cluster-name>
    PrincipalArn: <node-role-arn>
    Type: "EC2"
    AccessPolicies:
      - AccessScope:
          Type: cluster
          PolicyArn: arn:aws:eks::aws:cluster-access-policy/AmazonEKSAutoNodePolicy
    DependsOn: [ <cluster-name> ] # previously defined in CloudFormation

```

Untuk informasi tentang menerapkan CloudFormation tumpukan, lihat [Memulai CloudFormation](#)

Spesifikasi Kelas Node

```

apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: my-node-class
spec:
  # Required fields
  role: MyNodeRole # IAM role for EC2 instances

  subnetSelectorTerms:
    - tags:
        Name: "private-subnet"
        kubernetes.io/role/internal-elb: "1"
    # Alternative using direct subnet ID
    # - id: "subnet-0123456789abcdef0"

  securityGroupSelectorTerms:
    - tags:
        Name: "eks-cluster-sg"
    # Alternative approaches:
    # - id: "sg-0123456789abcdef0"
    # - name: "eks-cluster-security-group"

  # Optional: Pod subnet selector for advanced networking
  podSubnetSelectorTerms:
    - tags:
        Name: "pod-subnet"
        kubernetes.io/role/pod: "1"

```



```

# Alternative using direct subnet ID
# - id: "subnet-0987654321fedcba0"

# Optional: Selects on-demand capacity reservations and capacity blocks
# for EKS Auto Mode to prioritize.
capacityReservationSelectorTerms:
  - id: cr-56fac701cc1951b03
# Alternative Approaches
- tags:
    Name: "targeted-odcr"
# Optional owning account ID filter
owner: "012345678901"

# Optional fields
snatPolicy: Random # or Disabled

networkPolicy: DefaultAllow # or DefaultDeny
networkPolicyEventLogs: Disabled # or Enabled

ephemeralStorage:
  size: "80Gi" # Range: 1-59000Gi or 1-64000G or 1-58Ti or 1-64T
  iops: 3000 # Range: 3000-16000
  throughput: 125 # Range: 125-1000
# Optional KMS key for encryption
kmsKeyID: "arn:aws:kms:region:account:key/key-id"
# Accepted formats:
# KMS Key ID
# KMS Key ARN
# Key Alias Name
# Key Alias ARN

# Optional: Forward proxy, commonly requires certificateBundles as well
#for EC2, see https://repost.aws/knowledge-center/eks-http-proxy-containerd-
automation
advancedNetworking:
  httpsProxy: http://192.0.2.4:3128 #commonly port 3128 (Squid) or 8080 (NGINX) #Max
255 characters
#httpsProxy: http://[2001:db8::4]:3128 # IPv6 address with port, use []
noProxy: #Max 50 entries
  - localhost #Max 255 characters each
  - 127.0.0.1
  #- ::1 # IPv6 localhost
  #- 0:0:0:0:0:0:0:1 # IPv6 localhost

```

```

- 169.254.169.254 # EC2 Instance Metadata Service
#- [fd00:ec2::254] # IPv6 EC2 Instance Metadata Service
# Domains to exclude, put all VPC endpoints here
- .internal
- .eks.amazonaws.com

# Optional: Custom certificate bundles.
certificateBundles:
  - name: "custom-cert"
    data: "base64-encoded-cert-data"

# Optional: Additional EC2 tags (with restrictions)
tags:
  Environment: "production"
  Team: "platform"
# Note: Cannot use restricted tags like:
# - kubernetes.io/cluster/*
# - karpenter.sh/provisioner-name
# - karpenter.sh/nodepool
# - karpenter.sh/nodeclaim
# - karpenter.sh/managed-by
# - eks.amazonaws.com/nodeclass

```

Pertimbangan

- Enkripsi Volume - EKS menggunakan kunci KMS kustom yang dikonfigurasi untuk mengenkripsi volume root read-only dari instance dan volume data. read/write
- Ganti peran IAM node - Jika Anda mengubah peran IAM node yang terkait dengan `aNodeClass`, Anda harus membuat Entri Akses baru. EKS secara otomatis membuat Entri Akses untuk peran IAM node selama pembuatan cluster. Peran IAM node memerlukan Kebijakan Akses `AmazonEKSAutoNodePolicy` EKS. Untuk informasi selengkapnya, lihat [the section called “Berikan izin”](#).
- kepadatan pod maksimum - EKS membatasi jumlah maksimum pod pada node menjadi 110. Batas ini diterapkan setelah perhitungan pod maks yang ada. Untuk informasi selengkapnya, lihat [the section called “Jenis EC2 instans Amazon”](#).
- Tag - Jika Anda ingin menyebarkan tag dari Kubernetes ke Kubernetes EC2, Anda perlu mengonfigurasi izin IAM tambahan. Untuk informasi selengkapnya, lihat [the section called “Identitas dan akses”](#).

- Kelas node default - Jangan beri nama kelas node kustom `AndaDefault`. Ini karena Mode Otomatis EKS menyertakan `NodeClass` panggilan default yang secara otomatis disediakan ketika Anda mengaktifkan setidaknya satu built-in. `NodePool` Untuk informasi tentang mengaktifkan bawaan `NodePools`, lihat [the section called “Tinjau kumpulan node bawaan”](#).
- **`subnetSelectorTerms`** perilaku dengan beberapa subnet - Jika ada beberapa subnet yang cocok dengan `subnetSelectorTerms` kondisi atau yang Anda berikan oleh ID, Mode Otomatis EKS membuat node yang didistribusikan di seluruh subnet.
 - Jika subnet berada di Availability Zones (AZ) yang berbeda, Anda dapat menggunakan fitur Kubernetes seperti [kendala penyebaran topologi Pod dan Topology Aware Routing untuk menyebarkan pod dan lalu lintas](#) di seluruh zona.
 - Jika ada beberapa subnet dalam AZ yang sama yang cocok dengan `subnetSelectorTerms`, Mode Otomatis EKS membuat pod pada setiap node yang didistribusikan di seluruh subnet di AZ tersebut. Mode Otomatis EKS membuat antarmuka jaringan sekunder pada setiap node di subnet lain di AZ yang sama. Ini memilih berdasarkan jumlah alamat IP yang tersedia di setiap subnet, untuk menggunakan subnet lebih efisien. Namun, Anda tidak dapat menentukan subnet EKS Auto Mode mana yang digunakan untuk setiap pod; jika Anda membutuhkan pod untuk dijalankan di subnet tertentu, gunakan [the section called “Pemilihan subnet untuk Pod”](#) sebagai gantinya.

Pemilihan subnet untuk Pod

The `podSubnetSelectorTerms` and `podSecurityGroupSelectorTerms` Fields memungkinkan konfigurasi jaringan lanjutan dengan memungkinkan pod berjalan di subnet yang berbeda dari node mereka. Pemisahan ini memberikan kontrol yang ditingkatkan atas perutean lalu lintas jaringan dan kebijakan keamanan. Perhatikan bahwa `podSecurityGroupSelectorTerms` diperlukan dengan `podSubnetSelectorTerms`.

Kasus penggunaan

Gunakan `podSubnetSelectorTerms` saat Anda perlu:

- Pisahkan lalu lintas infrastruktur (node-to-node komunikasi) dari lalu lintas aplikasi (pod-to-pod komunikasi)
- Menerapkan konfigurasi jaringan yang berbeda ke subnet node daripada subnet pod.
- Menerapkan kebijakan keamanan atau aturan routing yang berbeda untuk node dan pod.

- Konfigurasi proxy terbalik atau pemfilteran jaringan khusus untuk lalu lintas node tanpa memengaruhi lalu lintas pod. Gunakan `advancedNetworking` dan `certificateBundles` untuk menentukan proxy terbalik Anda dan sertifikat yang ditandatangani sendiri atau pribadi untuk proxy.

Contoh konfigurasi

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: advanced-networking
spec:
  role: MyNodeRole

  # Subnets for EC2 instances (nodes)
  subnetSelectorTerms:
    - tags:
        Name: "node-subnet"
        kubernetes.io/role/internal-elb: "1"

  securityGroupSelectorTerms:
    - tags:
        Name: "eks-cluster-sg"

  # Separate subnets for pods
  podSubnetSelectorTerms:
    - tags:
        Name: "pod-subnet"
        kubernetes.io/role/pod: "1"

  podSecurityGroupSelectorTerms:
    - tags:
        Name: "eks-pod-sg"
```

Pertimbangan untuk memilih subnet untuk pod

- Mengurangi kepadatan pod: Pod yang lebih sedikit dapat berjalan pada setiap node saat menggunakan `podSubnetSelectorTerms`, karena antarmuka jaringan utama node berada di subnet node, dan tidak dapat digunakan untuk pod di subnet pod.
- Batasan memilih subnet: Standar `subnetSelectorTerms` dan `securityGroupSelectorTerms` konfigurasi tidak berlaku untuk pemilihan subnet pod.

- Perencanaan jaringan: Pastikan ruang alamat IP yang memadai di subnet node dan pod untuk mendukung kebutuhan beban kerja Anda.
- Konfigurasi routing: Verifikasi bahwa tabel rute dan Network Access Control List (ACL) dari subnet pod dikonfigurasi dengan benar untuk komunikasi antara subnet node dan pod.

Buat Node Pool untuk Mode Otomatis EKS

Kumpulan node Amazon EKS menawarkan cara yang fleksibel untuk mengelola sumber daya komputasi di kluster Kubernetes Anda. Topik ini menunjukkan cara membuat dan mengkonfigurasi kumpulan node dengan menggunakan Karpenter, alat penyediaan node yang membantu mengoptimalkan penskalaan cluster dan pemanfaatan sumber daya. Dengan NodePool sumber daya Karpenter, Anda dapat menentukan persyaratan spesifik untuk sumber daya komputasi Anda, termasuk jenis instans, zona ketersediaan, arsitektur, dan jenis kapasitas.

Anda tidak dapat memodifikasi kumpulan bawaan `system` dan `general-purpose` node. Anda hanya dapat mengaktifkan atau menonaktifkannya. Untuk informasi selengkapnya, lihat [the section called “Tinjau kumpulan node bawaan”](#).

NodePool Spesifikasi ini memungkinkan kontrol halus atas sumber daya komputasi kluster EKS Anda melalui berbagai label dan persyaratan yang didukung. Ini termasuk opsi untuk menentukan kategori EC2 instance, konfigurasi CPU, zona ketersediaan, arsitektur (ARM64/AMD64), dan jenis kapasitas (spot atau on-demand). Anda juga dapat menetapkan batas sumber daya untuk penggunaan CPU dan memori, memastikan kluster Anda tetap berada dalam batas operasional yang diperlukan.

Mode Otomatis EKS memanfaatkan label Kubernetes yang terkenal untuk memberikan cara yang konsisten dan terstandarisasi dalam mengidentifikasi karakteristik node. Label ini, seperti `topology.kubernetes.io/zone` untuk zona ketersediaan dan `kubernetes.io/arch` arsitektur CPU, mengikuti konvensi Kubernetes yang telah ditetapkan. Selain itu, label khusus EKS (diawali dengan `eks.amazonaws.com/`) memperluas fungsionalitas ini dengan atribut AWS-spesifik seperti tipe instance, produsen CPU, kemampuan GPU, dan spesifikasi jaringan. Sistem pelabelan standar ini memungkinkan integrasi tanpa batas dengan alat Kubernetes yang ada sambil menyediakan integrasi infrastruktur yang mendalam. AWS

Buat NodePool

Ikuti langkah-langkah berikut NodePool untuk membuat kluster Amazon EKS Anda:

1. Buat file YAMB bernama `nodepool1.yaml` dengan NodePool konfigurasi yang Anda butuhkan. Anda dapat menggunakan konfigurasi sampel di bawah ini.

2. Terapkan NodePool ke cluster Anda:

```
kubectl apply -f nodepool.yaml
```

3. Verifikasi bahwa NodePool telah berhasil dibuat:

```
kubectl get nodepools
```

4. (Opsional) Pantau NodePool status:

```
kubectl describe nodepool default
```

Pastikan NodePool referensi Anda valid NodeClass yang ada di cluster Anda. NodeClass Mendefinisikan konfigurasi AWS-spesifik untuk sumber daya komputasi Anda. Untuk informasi selengkapnya, lihat [the section called “Buat kelas simpul”](#).

Sampel NodePool

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: my-node-pool
spec:
  template:
    metadata:
      labels:
        billing-team: my-team
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default

      requirements:
        - key: "eks.amazonaws.com/instance-category"
          operator: In
          values: ["c", "m", "r"]
        - key: "eks.amazonaws.com/instance-cpu"
          operator: In
          values: ["4", "8", "16", "32"]
        - key: "topology.kubernetes.io/zone"
```

```

operator: In
values: ["us-west-2a", "us-west-2b"]
- key: "kubernetes.io/arch"
operator: In
values: ["arm64", "amd64"]

```

```

limits:
  cpu: "1000"
  memory: 1000Gi

```

Label yang Didukung Mode Otomatis EKS

Mode Otomatis EKS mendukung label terkenal berikut ini.

Note

Mode Otomatis EKS menggunakan label yang berbeda dari Karpenter. Label yang terkait dengan instance EC2 terkelola dimulai dengan `eks.amazonaws.com`.

Label	Contoh	Deskripsi
<code>topology.kubernetes.io/zone</code>	<code>us-timur-2a</code>	AWS wilayah
<code>node.kubernetes.io/instance-type</code>	<code>g4dn.8xlarge</code>	AWS jenis contoh
<code>kubernetes.io/lengkungan</code>	<code>amd64</code>	Arsitektur didefinisikan oleh nilai-nilai GOARCH pada instance
<code>karpenter.sh/tipe kapasitas</code>	<code>melihat</code>	Jenis kapasitas meliputi <code>spot</code> , <code>on-demand</code>
<code>eks.amazonaws.com/instance-hypervisor</code>	<code>nitro</code>	Jenis instans yang menggunakan hypervisor tertentu
<code>eks.amazonaws.com/compute-type</code>	<code>auto</code>	Mengidentifikasi node terkelola Mode Otomatis EKS

Label	Contoh	Deskripsi
eks.amazonaws.com/ -didukung instance-encryption-in-transit	true	Jenis instans yang mendukung (atau tidak) enkripsi dalam transit
eks.amazonaws.com/instance-category	g	Jenis instance dari kategori yang sama, biasanya string sebelum nomor generasi
eks.amazonaws.com/instance-generation	4	Nomor generasi tipe instans dalam kategori instance
eks.amazonaws.com/instance-family	g4dn	Jenis instance dari properti serupa tetapi jumlah sumber daya yang berbeda
eks.amazonaws.com/instance-size	8xlarge	Jenis contoh dari jumlah sumber daya yang sama tetapi properti yang berbeda
eks.amazonaws.com/instance-cpu	32	Jumlah CPUs pada instance
eks.amazonaws.com/ instance-cpu-manufacturer	aws	Nama produsen CPU
eks.amazonaws.com/instance-memory	131072	Jumlah mebibytes memori pada instance
eks.amazonaws.com/ instance-ebs-bandwidth	9500	Jumlah maksimum megabit EBS yang tersedia pada instans
eks.amazonaws.com/ instance-network-bandwidth	131072	Jumlah megabit dasar yang tersedia pada instance
eks.amazonaws.com/ instance-gpu-name	t4	Nama GPU pada instance, jika tersedia
eks.amazonaws.com/ instance-gpu-manufacturer	nvidia	Nama produsen GPU
eks.amazonaws.com/ instance-gpu-count	1	Jumlah GPUs pada instance

Label	Contoh	Deskripsi
eks.amazonaws.com/ instance-gpu-memory	16384	Jumlah mebibytes memori pada GPU
eks.amazonaws.com/ instance-local-nvme	900	Jumlah gibibyte penyimpanan nvme lokal pada instance

Note

Mode Otomatis EKS hanya mendukung instance tertentu, dan memiliki persyaratan ukuran minimum. Untuk informasi selengkapnya, lihat [the section called “Referensi instans yang didukung Mode Otomatis EKS”](#).

Mode Otomatis EKS Tidak Didukung Label

Mode Otomatis EKS tidak mendukung label berikut.

- Mode Otomatis EKS hanya mendukung Linux
 - `node.kubernetes.io/windows-build`
 - `kubernetes.io/os`

Nonaktifkan kolom node bawaan

Jika Anda membuat kumpulan node kustom, Anda dapat menonaktifkan kumpulan node bawaan. Untuk informasi selengkapnya, lihat [the section called “Tinjau kumpulan node bawaan”](#).

Cluster tanpa kolom node bawaan

Anda dapat membuat cluster tanpa kumpulan node bawaan. Ini sangat membantu ketika organisasi Anda telah membuat kumpulan node yang disesuaikan.

Note

Saat Anda membuat cluster tanpa kumpulan node bawaan, cluster tidak default NodeClass disediakan secara otomatis. Anda harus membuat kustom NodeClass. Untuk informasi selengkapnya, lihat [the section called “Buat kelas simpul”](#).

Ikhtisar:

1. Buat cluster EKS dengan keduanya nodePools dan nodeRoleArn nilai kosong.

- Contoh autoModeConfig eksctl:

```
autoModeConfig:
  enabled: true
  nodePools: []
  # Do not set a nodeRoleARN
```

Untuk informasi selengkapnya, lihat [the section called “CLI eksctl”](#)

2. Buat kelas node kustom dengan peran node ARN

- Untuk informasi selengkapnya, lihat [the section called “Buat kelas simpul”](#)

3. Buat entri akses untuk kelas node kustom

- Untuk informasi selengkapnya, lihat [the section called “Buat entri akses kelas node”](#)

4. Buat kumpulan node kustom, seperti dijelaskan di atas.

Gangguan

Anda dapat mengonfigurasi Mode Otomatis EKS untuk mengganggu Node melalui Anda NodePool dalam berbagai cara. Anda dapat menggunakan `spec.disruption.consolidationPolicy`, `spec.disruption consolidateAfter`, atau `spec.template.spec.expireAfter`. Anda juga dapat menilai batas gangguan Mode Otomatis EKS melalui NodePool's. `spec.disruption.budgets` Anda juga dapat mengontrol jendela waktu dan jumlah Node simultan yang terganggu. Untuk petunjuk tentang mengonfigurasi perilaku ini, lihat [Gangguan](#) dalam Dokumentasi Karpenter.

Anda dapat mengonfigurasi gangguan untuk kumpulan node menjadi:

- Identifikasi kapan instance kurang dimanfaatkan, dan konsolidasikan beban kerja.

- Buat anggaran gangguan kumpulan node untuk menilai penghentian node batas karena penyimpangan, kekosongan, dan konsolidasi.

Secara default, Mode Otomatis EKS:

- Mengkonsolidasikan instance yang kurang dimanfaatkan.
- Mengakhiri instance setelah 336 jam.
- Menetapkan anggaran gangguan tunggal 10% dari node.
- Memungkinkan Node diganti karena drift ketika AMI Mode Otomatis baru dirilis, yang terjadi kira-kira sekali per minggu.

Masa Tenggang Penghentian

Ketika `terminationGracePeriod` tidak didefinisikan secara eksplisit pada EKS Auto NodePool, sistem secara otomatis menerapkan masa tenggang penghentian 24 jam default ke yang terkait. NodeClaim Meskipun pelanggan EKS Auto tidak akan melihat `terminationGracePeriod` default dalam NodePool konfigurasi kustom mereka, mereka akan mengamati nilai default ini di NodeClaim. Fungsionalitas tetap konsisten apakah masa tenggang ditetapkan secara eksplisit pada NodePool atau default pada NodeClaim, memastikan perilaku penghentian node yang dapat diprediksi di seluruh cluster.

Buat IngressClass untuk mengkonfigurasi Application Load Balancer

Mode Otomatis EKS mengotomatiskan tugas rutin untuk penyeimbangan beban, termasuk mengekspos aplikasi cluster ke internet.

AWS menyarankan menggunakan Application Load Balancers (ALB) untuk melayani lalu lintas HTTP dan HTTPS. Application Load Balancers dapat merutekan permintaan berdasarkan konten permintaan. Untuk informasi selengkapnya tentang Application Load Balancer, lihat [Apa itu Elastic Load Balancing?](#)

EKS Auto Mode membuat dan mengkonfigurasi Application Load Balancers (). ALBs Misalnya, Mode Otomatis EKS membuat penyeimbang beban saat Anda membuat objek Ingress Kubernetes dan mengonfigurasinya untuk merutekan lalu lintas ke beban kerja klaster Anda.

Ikhtisar

1. Buat beban kerja yang ingin Anda ekspos ke internet.

2. Buat `IngressClassParams` sumber daya, tentukan nilai konfigurasi AWS tertentu seperti sertifikat yang akan digunakan untuk SSL/TLS dan Subnet VPC.
3. Buat `IngressClass` sumber daya, tentukan bahwa Mode Otomatis EKS akan menjadi pengontrol untuk sumber daya.
4. Buat `Ingress` sumber daya yang mengaitkan jalur HTTP dan port dengan beban kerja cluster.

Mode Otomatis EKS akan membuat Application Load Balancer yang menunjuk ke beban kerja yang ditentukan dalam `Ingress` sumber daya, menggunakan konfigurasi penyeimbang beban yang ditentukan dalam sumber daya. `IngressClassParams`

Prasyarat

- Mode Otomatis EKS Diaktifkan di Amazon EKS Cluster
- Kubectl dikonfigurasi untuk terhubung ke klaster Anda
 - Anda dapat menggunakan `kubectl apply -f <filename>` untuk menerapkan konfigurasi sampel file YAMAL di bawah ini ke cluster Anda.

Note

Mode Otomatis EKS memerlukan tag subnet untuk mengidentifikasi subnet publik dan pribadi.

Jika Anda membuat cluster Anda dengan `eksctl`, Anda sudah memiliki tag ini.

Pelajari cara [the section called “Tag subnet”](#).

Langkah 1: Buat beban kerja

Buat beban kerja yang ingin Anda ekspos ke internet. Ini bisa berupa sumber daya Kubernetes yang melayani lalu lintas HTTP, seperti Deployment atau Service. Untuk contoh ini, kami menggunakan layanan HTTP sederhana yang disebut `service-2048` yang melayani lalu lintas HTTP pada port 80.

Langkah 2: Buat `IngressClassParams`

Buat `IngressClassParams` objek untuk menentukan opsi konfigurasi AWS tertentu untuk Application Load Balancer. Dalam contoh ini, kami membuat `IngressClassParams`

sumber daya bernama alb (yang akan Anda gunakan pada langkah berikutnya) yang menentukan skema penyeimbang beban seperti internet-facing dalam file bernama `alb-ingressclassparams.yaml`

```
apiVersion: eks.amazonaws.com/v1
kind: IngressClassParams
metadata:
  name: alb
spec:
  scheme: internet-facing
```

Terapkan konfigurasi ke cluster Anda:

```
kubectl apply -f alb-ingressclassparams.yaml
```

Langkah 3: Buat IngressClass

Buat IngressClass yang mereferensikan nilai konfigurasi AWS tertentu yang ditetapkan dalam IngressClassParams sumber daya dalam file bernama `alb-ingressclass.yaml`. Perhatikan `namaIngressClass`. Dalam contoh ini, keduanya IngressClass dan IngressClassParams diberi nama `alb`.

Gunakan `is-default-class` anotasi untuk mengontrol apakah Ingress sumber daya harus menggunakan kelas ini secara default.

```
apiVersion: networking.k8s.io/v1
kind: IngressClass
metadata:
  name: alb
  annotations:
    # Use this annotation to set an IngressClass as Default
    # If an Ingress doesn't specify a class, it will use the Default
    ingressclass.kubernetes.io/is-default-class: "true"
spec:
  # Configures the IngressClass to use EKS Auto Mode
  controller: eks.amazonaws.com/alb
  parameters:
    apiGroup: eks.amazonaws.com
    kind: IngressClassParams
    # Use the name of the IngressClassParams set in the previous step
    name: alb
```

Untuk informasi selengkapnya tentang opsi konfigurasi, lihat [the section called “IngressClassParams Referensi”](#).

Terapkan konfigurasi ke cluster Anda:

```
kubectl apply -f alb-ingressclass.yaml
```

Langkah 4: Buat Ingress

Buat Ingress sumber daya dalam file bernama `alb-ingress.yaml`. Tujuan dari sumber daya ini adalah untuk mengaitkan path dan port pada Application Load Balancer dengan beban kerja di cluster Anda. Untuk contoh ini, kami membuat Ingress sumber daya bernama `2048-ingress` yang merutekan lalu lintas ke layanan bernama `service-2048` pada port 80.

Untuk informasi selengkapnya tentang mengonfigurasi resource ini, lihat [Ingress](#) di Dokumentasi Kubernetes.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: 2048-ingress
spec:
  # this matches the name of IngressClass.
  # this can be omitted if you have a default ingressClass in cluster: the one with
  ingressclass.kubernetes.io/is-default-class: "true"  annotation
  ingressClassName: alb
  rules:
    - http:
        paths:
          - path: /*
            pathType: ImplementationSpecific
            backend:
              service:
                name: service-2048
                port:
                  number: 80
```

Terapkan konfigurasi ke cluster Anda:

```
kubectl apply -f alb-ingress.yaml
```

Langkah 5: Periksa Status

Gunakan `kubectl` untuk menemukan status `Ingress`. Diperlukan beberapa menit agar penyeimbang beban tersedia.

Gunakan nama `Ingress` sumber daya yang Anda tetapkan pada langkah sebelumnya. Misalnya:

```
kubectl get ingress 2048-ingress
```

Setelah sumber daya siap, ambil nama domain penyeimbang beban.

```
kubectl get ingress 2048-ingress -o  
jsonpath='{.status.loadBalancer.ingress[0].hostname}'
```

Untuk melihat layanan di browser web, tinjau port dan jalur yang ditentukan dalam `Ingress` penyelamatan.

Langkah 6: Pembersihan

Untuk membersihkan penyeimbang beban, gunakan perintah berikut:

```
kubectl delete ingress 2048-ingress  
kubectl delete ingressclass alb  
kubectl delete ingressclassparams alb
```

Mode Otomatis EKS akan secara otomatis menghapus penyeimbang beban terkait di AWS akun Anda.

IngressClassParams Referensi

Tabel di bawah ini adalah referensi cepat untuk opsi konfigurasi yang umum digunakan.

Bidang	Deskripsi	Nilai contoh
scheme	Mendefinisikan apakah ALB bersifat internal atau berhadapan dengan internet	internet-facing

Bidang	Deskripsi	Nilai contoh
<code>namespaceSelector</code>	Membatasi ruang nama mana yang dapat menggunakan ini IngressClass	<code>environment: prod</code>
<code>group.name</code>	Kelompokkan beberapa Ingress untuk berbagi satu ALB	<code>retail-apps</code>
<code>ipAddressType</code>	Menetapkan jenis alamat IP untuk ALB	<code>dualstack</code>
<code>subnets.ids</code>	Daftar subnet IDs untuk penyebaran ALB	<code>subnet-xxxx, subnet-yyyy</code>
<code>subnets.tags</code>	Tag filter untuk memilih subnet untuk ALB	<code>Environment: prod</code>
<code>certificateARNs</code>	ARNs Sertifikat SSL untuk digunakan	<code>arn:aws:acm:region:account:certificate/id</code>
<code>tags</code>	Tag khusus untuk AWS sumber daya	<code>Environment: prod, Team: platform</code>
<code>loadBalancerAttributes</code>	Atribut khusus penyeimbang beban	<code>idle_timeout.timeout_seconds: 60</code>

Pertimbangan

- Anda tidak dapat menggunakan Anotasi pada IngressClass untuk mengonfigurasi penyeimbang beban dengan Mode Otomatis EKS.
- Anda harus memperbarui Peran IAM Cluster untuk mengaktifkan propagasi tag dari Kubernetes ke sumber daya Load AWS Balancer. Untuk informasi selengkapnya, lihat [the section called “AWS Tag khusus untuk sumber daya EKS Auto”](#).
- Untuk informasi tentang mengaitkan sumber daya dengan Mode Otomatis EKS atau Pengontrol Load AWS Balancer yang dikelola sendiri, lihat [the section called “Referensi migrasi”](#)

- Untuk informasi tentang memperbaiki masalah dengan penyeimbang beban, lihat. [the section called “Pemecahan Masalah”](#)
- Untuk pertimbangan lebih lanjut tentang penggunaan kemampuan load balancing dari Mode Otomatis EKS, lihat. [the section called “Penyeimbangan beban”](#)

Tabel berikut memberikan perbandingan rinci perubahan, anotasi Ingress IngressClassParams, dan TargetGroupBinding konfigurasi untuk Mode Otomatis EKS. Tabel ini menyoroti perbedaan utama antara kemampuan penyeimbangan beban Mode Otomatis EKS dan pengontrol penyeimbang beban sumber terbuka, termasuk perubahan versi API, fitur yang tidak digunakan lagi, dan nama parameter yang diperbarui.

IngressClassParams

Sebelumnya	Baru	Deskripsi
<code>elbv2.k8s.aws/v1beta1</code>	<code>eks.amazonaws.com/v1</code>	Perubahan versi API
<code>spec.certificateArn</code>	<code>spec.certificateARNs</code>	Support untuk beberapa sertifikat ARNs
<code>spec.subnets.tags</code>	<code>spec.subnets.matchTags</code>	Skema pencocokan subnet yang diubah
<code>spec.listeners.listenerAttributes</code>	<code>spec.listeners.attributes</code>	Penamaan atribut yang disederhanakan

Anotasi masuk

Sebelumnya	Baru	Deskripsi
<code>kubernetes.io/ingress.class</code>	Tidak didukung	Gunakan <code>spec.ingressClassName</code> pada objek Ingress
<code>alb.ingress.kubernetes.io/group.name</code>	Tidak didukung	Tentukan grup IngressClass hanya

Sebelumnya	Baru	Deskripsi
<code>alb.ingress.kubernetes.io/waf-acl-id</code>	Tidak didukung	Gunakan WAF v2 sebagai gantinya
<code>alb.ingress.kubernetes.io/web-acl-id</code>	Tidak didukung	Gunakan WAF v2 sebagai gantinya
<code>alb.ingress.kubernetes.io/shield-advanced-protection</code>	Tidak didukung	Integrasi Shield dinonaktifkan
<code>alb.ingress.kubernetes.io/auth-type: oidc</code>	Tidak didukung	Jenis Auth OIDC saat ini tidak didukung

TargetGroupBinding

Sebelumnya	Baru	Deskripsi
<code>elbv2.k8s.aws/v1beta1</code>	<code>eks.amazonaws.com/v1</code>	Perubahan versi API
<code>spec.targetType</code> opsional	<code>spec.targetType</code> diperlukan	Spesifikasi tipe target eksplisit
<code>spec.networking.ingress.from</code>	Tidak didukung	Tidak lagi mendukung NLB tanpa grup keamanan

Gunakan Anotasi Layanan untuk mengonfigurasi Network Load Balancers

Pelajari cara mengonfigurasi Network Load Balancers (NLB) di Amazon EKS menggunakan anotasi layanan Kubernetes. Topik ini menjelaskan anotasi yang didukung oleh Mode Otomatis EKS untuk menyesuaikan perilaku NLB, termasuk aksesibilitas internet, pemeriksaan kesehatan, SSL/TLS penghentian, dan mode penargetan IP.

Ketika Anda membuat layanan Kubernetes dari tipe `LoadBalancer` dalam Mode Otomatis EKS, EKS secara otomatis menyediakan dan mengonfigurasi AWS Network Load Balancer berdasarkan anotasi yang Anda tentukan. Pendekatan deklaratif ini memungkinkan Anda untuk mengelola konfigurasi penyeimbang beban secara langsung melalui manifes Kubernetes Anda, mempertahankan infrastruktur sebagai praktik kode.

Mode Otomatis EKS menangani penyediaan Network Load Balancer secara default untuk semua jenis layanan `LoadBalancer` - tidak diperlukan instalasi atau konfigurasi pengontrol tambahan. `loadBalancerClass: eks.amazonaws.com/nlb` Spesifikasi secara otomatis ditetapkan sebagai default cluster, merampingkan proses penerapan sambil mempertahankan kompatibilitas dengan beban kerja Kubernetes yang ada.

Note

Mode Otomatis EKS memerlukan tag subnet untuk mengidentifikasi subnet publik dan pribadi.

Jika Anda membuat cluster Anda dengan `eksctl`, Anda sudah memiliki tag ini.

Pelajari cara [the section called “Tag subnet”](#).

Layanan Sampel

Untuk informasi selengkapnya tentang Service sumber daya Kubernetes, lihat Dokumentasi Kubernetes.

Tinjau Service sumber daya sampel di bawah ini:

```
apiVersion: v1
kind: Service
metadata:
  name: echoserver
  annotations:
    # Specify the load balancer scheme as internet-facing to create a public-facing
    Network Load Balancer (NLB)
    service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing
spec:
  selector:
    app: echoserver
  ports:
    - port: 80
```

```
targetPort: 8080
protocol: TCP
type: LoadBalancer
# Specify the new load balancer class for NLB as part of EKS Auto Mode feature
# For clusters with Auto Mode enabled, this field can be omitted as it's the default
loadBalancerClass: eks.amazonaws.com/nlb
```

Anotasi yang umum digunakan

Tabel berikut mencantumkan anotasi yang umum digunakan yang didukung oleh Mode Otomatis EKS. Perhatikan bahwa Mode Otomatis EKS mungkin tidak mendukung semua anotasi.

Tip

Semua anotasi berikut harus diawali dengan `service.beta.kubernetes.io/`

Bidang	Deskripsi	Contoh
<code>aws-load-balancer-type</code>	Menentukan jenis penyeimbang beban. Gunakan <code>external</code> untuk penerapan baru.	<code>external</code>
<code>aws-load-balancer-nlb-target-type</code>	Menentukan apakah akan merutekan lalu lintas ke instance node atau langsung ke pod. IPs Gunakan <code>instance</code> untuk penerapan standar atau <code>ip</code> untuk routing pod langsung.	<code>instance</code>
<code>aws-load-balancer-scheme</code>	Mengontrol apakah penyeimbang beban internal atau menghadap ke internet.	<code>internet-facing</code>
<code>aws-load-balancer-healthcheck-protocol</code>	Protokol pemeriksaan kesehatan untuk kelompok	<code>HTTP</code>

Bidang	Deskripsi	Contoh
	sasaran. Opsi umum adalah TCP (default) atau HTTP.	
aws-load-balancer-healthcheck-path	Jalur HTTP untuk pemeriksaan kesehatan saat menggunakan HTTP/HTTPS protokol.	/healthz
aws-load-balancer-healthcheck-port	Pelabuhan yang digunakan untuk pemeriksaan kesehatan . Bisa berupa nomor port tertentu atau traffic-port .	traffic-port
aws-load-balancer-subnets	Menentukan subnet mana untuk membuat penyeimbang beban di. Dapat menggunakan subnet IDs atau nama.	subnet-xxxx, subnet-yyyy
aws-load-balancer-ssl-cert	ARN sertifikat SSL dari Certificate Manager untuk AWS HTTPS/TLS.	arn:aws:acm:region:account:certificate/cert-id
aws-load-balancer-ssl-ports	Menentukan port mana yang harus menggunakan SSL/TLS.	443, 8443
load-balancer-source-ranges	Rentang CIDR diizinkan untuk mengakses penyeimbang beban.	10.0.0.0/24, 192.168.1.0/24
aws-load-balancer-additional-resource-tags	AWS Tag tambahan untuk diterapkan ke penyeimbang beban dan sumber daya terkait.	Environment=prod, Team=platform
aws-load-balancer-ip-address-type	Menentukan apakah penyeimbang beban menggunakan IPv4 atau dual-stack (+). IPv4 IPv6	ipv4 atau dualstack

Pertimbangan

- Anda harus memperbarui Peran IAM Cluster untuk mengaktifkan propagasi tag dari Kubernetes ke sumber daya Load AWS Balancer. Untuk informasi selengkapnya, lihat [the section called “AWS Tag khusus untuk sumber daya EKS Auto”](#).
- Untuk informasi tentang mengaitkan sumber daya dengan Mode Otomatis EKS atau Pengontrol Load AWS Balancer yang dikelola sendiri, lihat. [the section called “Referensi migrasi”](#)
- Untuk informasi tentang memperbaiki masalah dengan penyeimbang beban, lihat. [the section called “Pemecahan Masalah”](#)
- Untuk pertimbangan lebih lanjut tentang penggunaan kemampuan load balancing dari Mode Otomatis EKS, lihat. [the section called “Penyeimbangan beban”](#)

Saat bermigrasi ke Mode Otomatis EKS untuk penyeimbangan beban, beberapa perubahan dalam anotasi layanan dan konfigurasi sumber daya diperlukan. Tabel berikut menguraikan perbedaan utama antara implementasi sebelumnya dan baru, termasuk opsi yang tidak didukung dan alternatif yang direkomendasikan.

Anotasi layanan

Sebelumnya	Baru	Deskripsi
<code>service.beta.kubernetes.io/load-balancer-source-ranges</code>	Tidak didukung	Gunakan <code>spec.loadBalancerSourceRanges</code> pada Layanan
<code>service.beta.kubernetes.io/aws-load-balancer-type</code>	Tidak didukung	Gunakan <code>spec.loadBalancerClass</code> pada Layanan
<code>service.beta.kubernetes.io/aws-load-balancer-internal</code>	Tidak didukung	Gunakan <code>service.beta.kubernetes.io/aws-load-balancer-scheme</code>
Berbagai atribut penyeimbang beban	Tidak didukung	Gunakan <code>service.beta.kubernetes.io/</code>

Sebelumnya	Baru	Deskripsi
		<code>aws-load-balancer-attributes</code>
<code>service.beta.kubernetes.io/aws-load-balancer-proxy-protocol</code>	Tidak didukung	Gunakan <code>service.beta.kubernetes.io/aws-load-balancer-attributes</code> sebagai gantinya
<code>service.beta.kubernetes.io/aws-load-balancer-access-log-enabled</code>	Tidak didukung	Gunakan <code>service.beta.kubernetes.io/aws-load-balancer-attributes</code> sebagai gantinya
<code>service.beta.kubernetes.io/aws-load-balancer-access-log-s3-bucket-name</code>	Tidak didukung	Gunakan <code>service.beta.kubernetes.io/aws-load-balancer-attributes</code> sebagai gantinya
<code>service.beta.kubernetes.io/aws-load-balancer-access-log-s3-bucket-prefix</code>	Tidak didukung	Gunakan <code>service.beta.kubernetes.io/aws-load-balancer-attributes</code> sebagai gantinya
<code>service.beta.kubernetes.io/aws-load-balancer-cross-zone-load-balancing-enabled</code>	Tidak didukung	Gunakan <code>service.beta.kubernetes.io/aws-load-balancer-attributes</code> sebagai gantinya

Untuk bermigrasi dari anotasi atribut load balancer yang tidak digunakan lagi, konsolidasikan setelan ini ke dalam anotasi `service.beta.kubernetes.io/aws-load-balancer-`

`attributes` Anotasi ini menerima daftar pasangan kunci-nilai yang dipisahkan koma untuk berbagai atribut penyeimbang beban. Misalnya, untuk menentukan protokol proxy, logging akses, dan penyeimbangan beban lintas zona, gunakan format berikut:

```
service.beta.kubernetes.io/aws-load-balancer-attributes: |
  access_logs.s3.enabled=true
  access_logs.s3.bucket=my-bucket
  access_logs.s3.prefix=my-prefix
  load_balancing.cross_zone.enabled=true
```

```
service.beta.kubernetes.io/aws-load-balancer-target-group-attributes: |
  proxy_protocol_v2.enabled=true
```

Format konsolidasi ini menyediakan cara yang lebih konsisten dan fleksibel untuk mengonfigurasi atribut penyeimbang beban sekaligus mengurangi jumlah anotasi individual yang diperlukan. Tinjau konfigurasi Layanan yang ada dan perbarui untuk menggunakan format gabungan ini.

TargetGroupBinding

Sebelumnya	Baru	Deskripsi
<code>elbv2.k8s.aws/v1beta1</code>	<code>eks.amazonaws.com/v1</code>	Perubahan versi API
<code>spec.targetType</code> opsional	<code>spec.targetType</code> diperlukan	Spesifikasi tipe target eksplisit
<code>spec.networking ingress.from</code>	Tidak didukung	Tidak lagi mendukung NLB tanpa grup keamanan

Buat kelas penyimpanan

A `StorageClass` di Amazon EKS Auto Mode mendefinisikan bagaimana volume Amazon EBS secara otomatis disediakan ketika aplikasi meminta penyimpanan persisten. Halaman ini menjelaskan cara membuat dan mengonfigurasi `StorageClass` yang berfungsi dengan Mode Otomatis Amazon EKS untuk menyediakan volume EBS.

Dengan mengonfigurasi `aStorageClass`, Anda dapat menentukan pengaturan default untuk volume EBS Anda termasuk jenis volume, enkripsi, IOPS, dan parameter penyimpanan lainnya. Anda juga dapat mengonfigurasi `StorageClass` untuk menggunakan kunci AWS KMS untuk manajemen enkripsi.

Mode Otomatis EKS tidak membuat `StorageClass` untuk Anda. Anda harus membuat `StorageClass` referensi `ebs.csi.eks.amazonaws.com` untuk menggunakan kemampuan penyimpanan Mode Otomatis EKS.

Pertama, buat file bernama `storage-class.yaml`:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: auto-ebs-sc
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
provisioner: ebs.csi.eks.amazonaws.com
volumeBindingMode: WaitForFirstConsumer
parameters:
  type: gp3
  encrypted: "true"
```

Kedua, terapkan kelas penyimpanan ke cluster Anda.

```
kubectl apply -f storage-class.yaml
```

Komponen kunci:

- `provisioner: ebs.csi.eks.amazonaws.com`- Menggunakan Mode Otomatis EKS
- `volumeBindingMode: WaitForFirstConsumer`- Menunda pembuatan volume hingga pod membutuhkannya
- `type: gp3`- Menentukan jenis volume EBS
- `encrypted: "true"`- EBS akan mengenkripsi volume apa pun yang dibuat menggunakan file. `StorageClass` EBS akan menggunakan alias `aws/ebs` kunci default. Untuk informasi selengkapnya, lihat [Cara kerja enkripsi Amazon EBS](#) di Panduan Pengguna Amazon EBS. Nilai ini opsional tetapi disarankan.
- `storageclass.kubernetes.io/is-default-class: "true"`- Kubernetes akan menggunakan kelas penyimpanan ini secara default, kecuali jika Anda menentukan kelas volume

yang berbeda pada klaim volume persisten. Nilai ini bersifat opsional. Berhati-hatilah saat menyetel nilai ini jika Anda bermigrasi dari pengontrol penyimpanan yang berbeda.

Gunakan kunci KMS yang dikelola sendiri untuk mengenkripsi volume EBS

Untuk menggunakan kunci KMS yang dikelola sendiri untuk mengenkripsi volume EBS yang diotomatisasi oleh Mode Otomatis EKS, Anda perlu:

1. Buat kunci KMS yang dikelola sendiri.
 - Untuk informasi selengkapnya, lihat [Membuat kunci KMS enkripsi simetris atau Cara Amazon Elastic Block Store \(Amazon EBS\) menggunakan KMS](#) di Panduan Pengguna KMS.
2. Buat kebijakan baru yang mengizinkan akses ke kunci KMS.
 - Gunakan contoh kebijakan IAM di bawah ini untuk membuat kebijakan. Masukkan ARN dari kunci KMS yang dikelola sendiri yang baru. Untuk informasi selengkapnya, lihat [Membuat peran dan melampirkan kebijakan \(konsol\)](#) di Panduan Pengguna AWS IAM.
3. Lampirkan kebijakan ke Peran Kluster EKS.
 - Gunakan AWS konsol untuk menemukan ARN dari EKS Cluster Role. Informasi peran terlihat di bagian Ikhtisar. Untuk informasi selengkapnya, lihat [the section called "IAM role klaster"](#).
4. Perbarui StorageClass untuk mereferensikan ID Kunci KMS di `parameters.kmsKeyId` lapangan.

Contoh Kebijakan IAM KMS yang dikelola sendiri

Perbarui nilai-nilai berikut dalam kebijakan di bawah ini:

- `<account-id>` ID AWS akun Anda, seperti 111122223333
- `<aws-region>`— AWS Wilayah cluster Anda, seperti us-west-2

```
{
  "Version": "2012-10-17",
  "Id": "key-auto-policy-3",
  "Statement": [
    {
      "Sid": "Enable IAM User Permissions",
      "Effect": "Allow",
      "Principal": {
```

```

        "AWS": "arn:aws:iam::<account-id>:root"
    },
    "Action": "kms:*",
    "Resource": "*"
  },
  {
    "Sid": "Allow access through EBS for all principals in the account that are
authorized to use EBS",
    "Effect": "Allow",
    "Principal": {
      "AWS": "*"
    },
    "Action": [
      "kms:Encrypt",
      "kms:Decrypt",
      "kms:ReEncrypt*",
      "kms:GenerateDataKey*",
      "kms:CreateGrant",
      "kms:DescribeKey"
    ],
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "kms:CallerAccount": "<account-id>",
        "kms:ViaService": "ec2.<aws-region>.amazonaws.com"
      }
    }
  }
]
}

```

Contoh KMS yang dikelola sendiri **StorageClass**

```

parameters:
  type: gp3
  encrypted: "true"
  kmsKeyId: <custom-key-arn>

```

StorageClassParameter Referensi

Untuk informasi umum tentang StorageClass sumber daya Kubernetes, lihat [Kelas Penyimpanan](#) di Dokumentasi Kubernetes.

The parameters bagian dari StorageClass sumber daya khusus untuk AWS. Gunakan tabel berikut untuk meninjau opsi yang tersedia.

Parameter	Nilai	Default	Deskripsi
"csi.storage.k8s.io/fstype"	xfs, ext2, ext3, ext4	ext4	Jenis sistem file yang akan diformat selama pembuatan volume. Parameter ini peka huruf besar/kecil!
"ketik"	io1, io2, gp2, gp3, sc1, st1, standar, sbp1, sbg1	gp3	Jenis volume EBS.
"iOpsPerGB"			Operasi I/O per detik per GiB. Dapat ditentukan untuk IO1, IO2, dan GP3 volume.
"AllowAuto IOPSPerGBIncrease"	benar, salah	false	Ketika "true", driver CSI meningkatkan IOPS untuk volume ketika <code>iopsPerGB * <volume size></code> terlalu rendah untuk masuk ke dalam rentang IOPS yang didukung oleh AWS. Hal ini memungkinkan penyediaan dinamis untuk selalu berhasil, bahkan ketika pengguna menentukan kapasitas atau nilai PVC yang terlalu kecil. <code>iopsPerGB</code> Di sisi lain, ini

Parameter	Nilai	Default	Deskripsi
			mungkin menimbulkan biaya tambahan, karena volume tersebut memiliki IOPS yang lebih tinggi daripada yang diminta. <code>iopsPerGB</code>
<code>"iops"</code>			Operasi I/O per detik. Dapat ditentukan untuk IO1, IO2, dan GP3 volume.
<code>"throughput"</code>		125	Throughput dalam MIB/s. Hanya efektif ketika tipe volume gp3 ditentukan.
<code>"dienkripsi"</code>	benar, salah	false	Apakah volume harus dienkrpsi atau tidak. Nilai yang valid adalah "benar" atau "salah".
<code>"BlockExpress"</code>	benar, salah	false	Memungkinkan pembuatan volume Blok Ekspres io2.

Parameter	Nilai	Default	Deskripsi
"kmsKeyId"			ARN penuh dari kunci untuk digunakan saat mengenkripsi volume. Jika tidak ditentukan, AWS akan menggunakan kunci KMS default untuk wilayah volume berada. Ini akan menjadi kunci yang dibuat secara otomatis yang disebut /aws/ebs jika tidak diubah.
"BlockSize"			Ukuran blok yang akan digunakan saat memformat sistem file yang mendasarinya. Hanya didukung pada node linux dan dengan fstype ext2,, ext3 ext4, atau. xfs
"InodeSize"			Ukuran inode yang digunakan saat memformat sistem file yang mendasarinya. Hanya didukung pada node linux dan dengan fstype ext2,, ext3 ext4, atau. xfs

Parameter	Nilai	Default	Deskripsi
"bytesPerInode"			Yang bytes-per-inode digunakan saat memformat sistem file yang mendasarinya. Hanya didukung pada node linux dan dengan fstype ext2,,ext3. ext4
"numberOfInodes"			Yang number-of-inodes digunakan saat memformat sistem file yang mendasarinya. Hanya didukung pada node linux dan dengan fstype ext2,,ext3. ext4
"BigAllocext4"	benar, salah	false	Mengubah sistem ext4 file untuk menggunakan alokasi blok berkerumun dengan mengaktifkan opsi pemformatan. bigalloc Peringatan: bigalloc mungkin tidak sepenuhnya didukung dengan kernel Linux node Anda.

Parameter	Nilai	Default	Deskripsi
"ClusterSizeext4"			Ukuran cluster yang akan digunakan saat memformat sistem ext4 file saat fitur diaktifkan. bigalloc Catatan: ext4BigAlloc Parameter harus diatur ke true.

Untuk informasi selengkapnya, lihat [Driver AWS EBS CSI](#) di GitHub

Pertimbangan

Kemampuan penyimpanan blok EKS Auto Mode berbeda dari EBS CSI Driver.

- Penyediaan Statis
 - Jika Anda ingin menggunakan volume EBS yang dibuat secara eksternal dengan Mode Otomatis EKS, Anda perlu menambahkan AWS tag secara manual dengan kunci `eks:eks-cluster-name` dan nilai nama cluster.
- Noda Startup Node
 - Anda tidak dapat menggunakan fitur taint startup node untuk mencegah penjadwalan pod sebelum kesiapan kemampuan penyimpanan
- Tag Kustom pada Volume yang Disediakan Secara Dinamis
 - Anda tidak dapat menggunakan flag CLI tag ekstra untuk mengonfigurasi tag khusus pada volume EBS yang disediakan secara dinamis
 - Anda dapat menggunakan StorageClass penandaan untuk menambahkan tag khusus. Mode Otomatis EKS akan menambahkan tag ke AWS sumber daya terkait. Anda perlu memperbarui Peran IAM Cluster untuk tag khusus. Untuk informasi selengkapnya, lihat [the section called "AWS Tag khusus untuk sumber daya EKS Auto"](#).
- Metrik Kinerja Terperinci EBS
 - Anda tidak dapat mengakses metrik Prometheus untuk performa rinci EBS

Instal add-on CSI Snapshot Controller

Mode Otomatis EKS kompatibel dengan add-on CSI Snapshot Controller Amazon EKS.

AWS menyarankan Anda mengonfigurasi add-on ini untuk berjalan di kumpulan system node bawaan.

Untuk informasi selengkapnya, lihat:

- [the section called “Jalankan add-on penting”](#)
- [the section called “Tinjau kumpulan node bawaan”](#)
- [the section called “Pengontrol snapshot CSI”](#)

Untuk menginstal pengontrol snapshot di kumpulan node sistem

1. Buka kluster EKS Anda di AWS konsol
2. Dari tab Add-ons, pilih Dapatkan lebih banyak add-on
3. Pilih CSI Snapshot Controller dan kemudian Berikutnya
4. Pada halaman Konfigurasi pengaturan add-on yang dipilih, pilih Pengaturan konfigurasi opsional untuk melihat skema konfigurasi Add-on
 - a. Masukkan yaml berikut untuk mengaitkan pengontrol snapshot dengan kumpulan system node. Pengontrol snapshot mencakup toleransi terhadap noda. `CriticalAddonsOnly`

```
{
  "nodeSelector": {
    "karpenter.sh/nodepool": "system"
  }
}
```

- b. Pilih Berikutnya

5. Tinjau konfigurasi add-on dan kemudian pilih Buat

Nonaktifkan Mode Otomatis EKS

Anda dapat menonaktifkan Mode Otomatis EKS pada Kluster EKS yang ada. Ini adalah operasi yang merusak.

- EKS akan menghentikan semua EC2 instans yang dioperasikan oleh Mode Otomatis EKS.

- EKS akan menghapus semua Load Balancer yang dioperasikan oleh EKS Auto Mode.
- EKS tidak akan menghapus volume EBS yang disediakan oleh Mode Otomatis EKS.

Mode Otomatis EKS dirancang untuk sepenuhnya mengelola sumber daya yang dibuatnya. Intervensi manual dapat mengakibatkan Mode Otomatis EKS gagal membersihkan sumber daya tersebut sepenuhnya saat dinonaktifkan. Misalnya, jika Anda merujuk ke Grup Keamanan terkelola dari aturan Grup Keamanan eksternal, dan lupa untuk menghapus referensi tersebut sebelum Anda menonaktifkan Mode Otomatis EKS untuk kluster, Grup Keamanan terkelola akan bocor (tidak dihapus). Langkah-langkah di bawah ini menjelaskan cara menghapus Grup Keamanan yang bocor jika itu terjadi.

Nonaktifkan Mode Otomatis EKS (AWS Konsol)

1. Buka halaman ikhtisar cluster Anda di AWS Management Console.
2. Di bawah Mode Otomatis EKS pilih Kelola
3. Alihkan Mode Otomatis EKS ke. off

Jika ada Grup Keamanan terkelola yang tidak dihapus pada akhir proses ini, Anda dapat menghapusnya secara manual menggunakan deskripsi dari [Hapus grup keamanan](#).

Nonaktifkan Mode Otomatis EKS (AWS CLI)

Gunakan perintah berikut untuk menonaktifkan Mode Otomatis EKS pada cluster yang ada.

Anda harus menginstal aws CLI, dan masuk dengan izin yang cukup untuk mengelola kluster EKS. Untuk informasi selengkapnya, lihat [Penyiapan](#).

Note

Kemampuan komputasi, penyimpanan blok, dan penyeimbangan beban semuanya harus diaktifkan atau dinonaktifkan dalam permintaan yang sama.

```
aws eks update-cluster-config \
  --name $CLUSTER_NAME \
  --compute-config enabled=false \
  --kubernetes-network-config '{"elasticLoadBalancing":{"enabled": false}}' \
  --storage-config '{"blockStorage":{"enabled": false}}'
```

Anda dapat memeriksa apakah Grup Keamanan Mode Otomatis EKS yang bocor gagal dihapus setelah menonaktifkan Mode Otomatis EKS sebagai berikut:

```
aws ec2 describe-security-groups \
  --filters Name=tag:eks:eks-cluster-name,Values=<cluster-Name> Name=tag-
  key,Values=ingress.eks.amazonaws.com/resource,service.eks.amazonaws.com/resource --
  query "SecurityGroups[*].[GroupName]"
```

Untuk kemudian menghapus Grup Keamanan:

```
aws ec2 delete-security-group --group-name=<sg-name>
```

Memperbarui versi Kubernetes dari kluster Mode Otomatis EKS

Topik ini menjelaskan cara memperbarui versi Kubernetes dari kluster Mode Otomatis Anda. Mode Otomatis menyederhanakan proses pembaruan versi dengan menangani koordinasi pembaruan bidang kontrol dengan penggantian node, sambil mempertahankan ketersediaan beban kerja melalui anggaran gangguan pod.

Saat memutakhirkan kluster Mode Otomatis, banyak komponen yang secara tradisional memerlukan pembaruan manual sekarang dikelola sebagai bagian dari layanan. Memahami aspek otomatis dari proses peningkatan dan tanggung jawab Anda membantu memastikan transisi versi yang lancar untuk kluster Anda.

Pelajari tentang pembaruan dengan Mode Otomatis EKS

Setelah Anda memulai upgrade control plane, EKS Auto Mode akan meng-upgrade node di cluster Anda. Saat node kedaluwarsa, Mode Otomatis EKS akan menggantinya dengan node baru. Node baru memiliki versi Kubernetes baru yang sesuai. Mode Otomatis EKS mengamati anggaran gangguan pod saat memutakhirkan node.

Selain itu, Anda tidak perlu lagi memperbarui komponen seperti:

- Amazon VPC CNI
- AWS Pengontrol Load Balancer
- CoreDNS
- kube-proxy
- Karpenter

- AWS Pengemudi EBS CSI

Mode Otomatis EKS menggantikan komponen ini dengan fungsionalitas layanan.

Anda masih bertanggung jawab untuk memperbarui:

- Aplikasi dan beban kerja yang diterapkan ke kluster Anda
- Pengaya dan pengontrol yang dikelola sendiri
- Pengaya Amazon EKS
 - Pelajari cara [the section called “Perbarui add-on”](#)

Pelajari [Praktik Terbaik untuk Upgrade Cluster](#)

Mulai Pembaruan Cluster

Untuk memulai pembaruan cluster, lihat [the section called “Perbarui versi Kubernetes”](#).

Aktifkan atau Nonaktifkan Built-in NodePools

Mode Otomatis EKS memiliki dua built-in NodePools. Anda dapat mengaktifkan atau menonaktifkannya NodePools menggunakan AWS konsol, CLI, atau API.

NodePool Referensi bawaan

- `system`
 - Ini NodePool memiliki `CriticalAddonsOnly` noda. Banyak add-on EKS, seperti CoreDNS, mentolerir noda ini. Gunakan kumpulan node sistem ini untuk memisahkan aplikasi kluster-kritis.
 - Mendukung keduanya amd64 dan arm64 arsitektur.
- `general-purpose`
 - Ini NodePool memberikan dukungan untuk meluncurkan node untuk beban kerja tujuan umum di klaster Anda.
 - Hanya menggunakan amd64 arsitektur.

Keduanya built-in NodePools:

- Gunakan EKS default NodeClass
- Gunakan hanya kapasitas sesuai permintaan EC2

- Gunakan keluarga EC2 instance C, M, dan R
- Memerlukan generasi 5 atau instance yang lebih baru EC2

Note

Mengaktifkan setidaknya satu built-in NodePool diperlukan untuk EKS untuk menyediakan “default”. NodeClass Jika Anda menonaktifkan semua bawaan NodePools, Anda harus membuat kustom NodeClass dan mengonfigurasi a NodePool untuk menggunakannya. Untuk informasi lebih lanjut tentang NodeClasses, lihat [the section called “Buat kelas simpel”](#).

Prosedur

Prasyarat

- Versi terbaru dari AWS Command Line Interface (AWS CLI) diinstal dan dikonfigurasi pada perangkat Anda. Untuk memeriksa versi Anda saat ini, gunakan `aws --version`. Untuk menginstal versi terbaru, lihat [Menginstal](#) dan [Konfigurasi cepat](#) dengan `aws configure` di Panduan Pengguna Antarmuka Baris AWS Perintah.
- Masuk ke CLI dengan izin IAM yang cukup untuk membuat AWS sumber daya termasuk Kebijakan IAM, Peran IAM, dan Kluster EKS.

Aktifkan dengan AWS CLI

Gunakan perintah berikut untuk mengaktifkan keduanya bawaan NodePools:

```
aws eks update-cluster-config \
  --name <cluster-name> \
  --compute-config '{
    "nodeRoleArn": "<node-role-arn>",
    "nodePools": ["general-purpose", "system"],
    "enabled": true
  }' \
  --kubernetes-network-config '{
    "elasticLoadBalancing":{"enabled": true}
  }' \
  --storage-config '{
    "blockStorage":{"enabled": true}
```

```
}'
```

Anda dapat memodifikasi perintah untuk mengaktifkan secara selektif. NodePools

Nonaktifkan dengan AWS CLI

Gunakan perintah berikut untuk menonaktifkan keduanya bawaan NodePools:

```
aws eks update-cluster-config \
  --name <cluster-name> \
  --compute-config '{
    "enabled": true,
    "nodePools": []
  }' \
  --kubernetes-network-config '{
    "elasticLoadBalancing":{"enabled": true}}' \
  --storage-config '{
    "blockStorage":{"enabled": true}
  }'
```

Kontrol jika beban kerja diterapkan pada node Mode Otomatis EKS

Saat menjalankan beban kerja di kluster EKS dengan Mode Otomatis EKS, Anda mungkin perlu mengontrol apakah beban kerja tertentu berjalan pada node Mode Otomatis EKS atau jenis komputasi lainnya. Topik ini menjelaskan cara menggunakan pemilih node dan aturan afinitas untuk memastikan beban kerja Anda dijadwalkan pada infrastruktur komputasi yang dimaksud.

Contoh dalam topik ini menunjukkan cara menggunakan `eks.amazonaws.com/compute-type` label untuk meminta atau mencegah penyebaran beban kerja pada node Mode Otomatis EKS. Ini sangat berguna dalam cluster mode campuran tempat Anda menjalankan Mode Otomatis EKS dan jenis komputasi lainnya, seperti penyedia Karpenter yang dikelola sendiri atau Grup Node Terkelola EKS.

Node Mode Otomatis EKS telah menetapkan nilai label `eks.amazonaws.com/compute-type` ke `auto`. Anda dapat menggunakan label ini untuk mengontrol jika beban kerja diterapkan ke node yang dikelola oleh Mode Otomatis EKS.

Memerlukan beban kerja diterapkan ke node Mode Otomatis EKS

Note

`nodeSelector` ini tidak diperlukan untuk Mode Otomatis EKS. `nodeSelector` ini hanya relevan jika Anda menjalankan cluster dalam mode campuran, tipe node tidak dikelola oleh Mode Otomatis EKS. Misalnya, Anda mungkin memiliki kapasitas komputasi statis yang diterapkan ke kluster Anda dengan Grup Node Terkelola EKS, dan memiliki kapasitas komputasi dinamis yang dikelola oleh Mode Otomatis EKS.

Anda dapat menambahkan ini `nodeSelector` ke Deployment atau beban kerja lainnya agar Kubernetes menjadwalkannya ke node Mode Otomatis EKS.

```
apiVersion: apps/v1
kind: Deployment
spec:
  template:
    nodeSelector:
      eks.amazonaws.com/compute-type: auto
```

Memerlukan beban kerja tidak diterapkan ke node Mode Otomatis EKS

Anda dapat menambahkan ini `nodeAffinity` ke Deployment atau beban kerja lainnya agar Kubernetes tidak menjadwalkannya ke node Mode Otomatis EKS.

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: eks.amazonaws.com/compute-type
          operator: NotIn
          values:
            - auto
```

Jalankan add-on penting pada instance khusus

Dalam topik ini, Anda akan belajar bagaimana menerapkan beban kerja dengan `CriticalAddonsOnly` toleransi sehingga Mode Otomatis EKS akan menjadwalkannya ke kumpulan node `system`

Kumpulan `system` node bawaan EKS Auto Mode dirancang untuk menjalankan add-on penting pada instance khusus. Pemisahan ini memastikan komponen penting memiliki sumber daya khusus dan terisolasi dari beban kerja umum, meningkatkan stabilitas dan kinerja kluster secara keseluruhan.

Panduan ini menunjukkan cara menerapkan add-on ke kumpulan `system` node dengan memanfaatkan `CriticalAddonsOnly` toleransi dan pemilih node yang sesuai. Dengan mengikuti langkah-langkah ini, Anda dapat memastikan bahwa aplikasi penting Anda dijadwalkan ke `system` node khusus, memanfaatkan manfaat isolasi dan alokasi sumber daya yang disediakan oleh struktur kumpulan node khusus EKS Auto Mode.

Mode Otomatis EKS memiliki dua kumpulan node bawaan: `general-purpose` dan `system`. Untuk informasi selengkapnya, lihat [the section called “Tinjau kumpulan node bawaan”](#).

Tujuan dari kumpulan `system` node adalah untuk memisahkan add-on kritis ke node yang berbeda. Node yang disediakan oleh kumpulan `system` node memiliki taint `CriticalAddonsOnly` Kubernetes. Kubernetes hanya akan menjadwalkan pod ke node ini jika mereka memiliki toleransi yang sesuai. Untuk informasi selengkapnya, lihat [Taints and Tolerations dalam dokumentasi Kubernetes](#).

Prasyarat

- EKS Auto Mode Cluster dengan kumpulan `system` node bawaan diaktifkan. Untuk informasi selengkapnya, silakan lihat [the section called “Tinjau kumpulan node bawaan”](#)
- `kubectl` diinstal dan dikonfigurasi. Untuk informasi selengkapnya, lihat [Penyiapan](#).

Prosedur

Tinjau contoh `yaml` di bawah ini. Perhatikan konfigurasi berikut:

- `nodeSelector`— Ini mengaitkan beban kerja dengan kumpulan `system` node bawaan. Kumpulan node ini harus diaktifkan dengan AWS API. Untuk informasi selengkapnya, lihat [the section called “Tinjau kumpulan node bawaan”](#).

- **tolerations**—Toleransi ini mengatasi `CriticalAddonsOnly` noda pada node di kumpulan `node.system`

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sample-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: sample-app
  template:
    metadata:
      labels:
        app: sample-app
    spec:
      nodeSelector:
        karpenter.sh/nodepool: system
      tolerations:
        - key: "CriticalAddonsOnly"
          operator: "Exists"
      containers:
        - name: app
          image: nginx:latest
          resources:
            requests:
              cpu: "500m"
              memory: "512Mi"
```

Untuk memperbarui beban kerja agar berjalan di kumpulan `system` node, Anda perlu:

1. Perbarui beban kerja yang ada untuk menambahkan konfigurasi berikut yang dijelaskan di atas:
 - `nodeSelector`
 - `tolerations`
2. Terapkan beban kerja yang diperbarui ke cluster Anda dengan `kubectl apply`

Setelah memperbarui beban kerja, itu akan berjalan pada node khusus.

Gunakan Kebijakan Jaringan dengan Mode Otomatis EKS

Kebijakan jaringan memungkinkan Anda mengontrol arus lalu lintas di alamat IP atau tingkat port dalam kluster Amazon EKS Anda. Topik ini menjelaskan cara mengaktifkan dan menggunakan kebijakan jaringan dengan Mode Otomatis EKS.

Prasyarat

- Cluster Amazon EKS dengan Mode Otomatis EKS diaktifkan
- kubectl dikonfigurasi untuk terhubung ke kluster Anda

Langkah 1: Aktifkan Pengontrol Kebijakan Jaringan

Untuk menggunakan kebijakan jaringan dengan Mode Otomatis EKS, pertama-tama Anda harus mengaktifkan Network Policy Controller dengan menerapkan a ConfigMap ke cluster Anda.

1. Buat file bernama `enable-network-policy.yaml` dengan konten berikut:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: amazon-vpc-cni
  namespace: kube-system
data:
  enable-network-policy-controller: "true"
```

2. Terapkan ConfigMap ke cluster Anda:

```
kubectl apply -f enable-network-policy.yaml
```

Langkah 2: Aktifkan Kebijakan Jaringan di Kelas Node

Sebelum Anda dapat menggunakan kebijakan jaringan, Anda perlu memastikan bahwa Kelas Node Anda dikonfigurasi untuk mendukungnya. Ikuti langkah-langkah ini:

1. Membuat atau mengedit file YAMLKelas Node (misalnya, `nodeclass-network-policy.yaml`) dengan konten berikut:

```
apiVersion: eks.amazonaws.com/v1
```

```
kind: NodeClass
metadata:
  name: network-policy-enabled
spec:
  # Enables network policy support
  networkPolicy: DefaultAllow
  # Optional: Enables logging for network policy events
  networkPolicyEventLogs: Enabled
  # Include other Node Class configurations as needed
```

2. Menerapkan konfigurasi Node Class ke cluster Anda:

```
kubectl apply -f nodeclass-network-policy.yaml
```

3. Verifikasi bahwa Kelas Node telah dibuat:

```
kubectl get nodeclass network-policy-enabled
```

4. Perbarui Node Pool Anda untuk menggunakan Kelas Node ini. Untuk informasi selengkapnya, lihat [the section called “Buat kumpulan simpul”](#).

Setelah node Anda menggunakan Kelas Node ini, mereka akan dapat menegakkan kebijakan jaringan. Anda sekarang dapat melanjutkan untuk membuat dan menerapkan kebijakan jaringan untuk mengontrol lalu lintas dalam klaster Anda. Untuk semua opsi konfigurasi kelas node, lihat [the section called “Buat kelas simpul”](#).

Langkah 3: Buat dan uji kebijakan jaringan

Kluster Mode Otomatis EKS Anda sekarang dikonfigurasi untuk mendukung kebijakan jaringan Kubernetes. Anda dapat menguji ini dengan [the section called “Demo kebijakan Stars”](#).

Tag subnet untuk Mode Otomatis EKS

Jika Anda menggunakan kemampuan load balancing dari Mode Otomatis EKS, Anda perlu menambahkan AWS tag ke subnet VPC Anda.

Latar Belakang

Tag ini mengidentifikasi subnet yang terkait dengan cluster, dan yang lebih penting jika subnet bersifat publik atau pribadi.

Subnet publik memiliki akses internet langsung melalui gateway internet. Mereka digunakan untuk sumber daya yang perlu diakses publik seperti penyeimbang beban.

Subnet pribadi tidak memiliki akses internet langsung dan menggunakan gateway NAT untuk lalu lintas keluar. Mereka digunakan untuk sumber daya internal seperti node EKS yang tidak membutuhkan publik IPs.

Untuk mempelajari selengkapnya tentang gateway NAT dan gateway Internet, lihat [Menyambungkan VPC ke jaringan lain](#) di Panduan Pengguna Amazon Virtual Private Cloud (VPC).

Persyaratan

Pada saat ini, subnet yang digunakan untuk penyeimbangan beban oleh Mode Otomatis EKS diharuskan memiliki salah satu tag berikut.

Subnet publik

Subnet publik digunakan untuk penyeimbang beban yang menghadap ke internet. Subnet ini harus memiliki tag berikut:

Kunci	Nilai
<code>kubernetes.io/role/elb</code>	1 atau ``

Subnet privat

Subnet pribadi digunakan untuk penyeimbang beban internal. Subnet ini harus memiliki tag berikut:

Kunci	Nilai
<code>kubernetes.io/role/internal-elb</code>	1 atau ``

Prosedur

Sebelum Anda mulai, identifikasi subnet mana yang bersifat publik (dengan akses Internet Gateway) dan mana yang pribadi (menggunakan NAT Gateway). Anda memerlukan izin untuk memodifikasi sumber daya VPC.

AWS Management Console

1. Buka konsol Amazon VPC dan navigasikan ke Subnet
2. Pilih subnet yang akan diberi tag
3. Pilih tab Tag dan pilih Tambahkan tag
4. Tambahkan tag yang sesuai:
 - Untuk subnet publik: Key= `kubernetes.io/role/elb`
 - Untuk subnet pribadi: Key= `kubernetes.io/role/internal-elb`
5. Tetapkan Nilai ke 1 atau biarkan kosong
6. Simpan dan ulangi untuk subnet yang tersisa

AWS CLI

Untuk subnet publik:

```
aws ec2 create-tags \
  --resources subnet-ID \
  --tags Key=kubernetes.io/role/elb,Value=1
```

Untuk subnet pribadi:

```
aws ec2 create-tags \
  --resources subnet-ID \
  --tags Key=kubernetes.io/role/internal-elb,Value=1
```

Ganti subnet-ID dengan subnet ID Anda yang sebenarnya.

Menerapkan beban kerja yang dipercepat

Tutorial ini menunjukkan bagaimana Amazon EKS Auto Mode menyederhanakan peluncuran beban kerja yang dipercepat. Amazon EKS Auto Mode merampingkan operasi di luar kluster itu sendiri dengan mengotomatiskan komponen infrastruktur utama yang menyediakan kemampuan komputasi, jaringan, penyeimbangan beban, penyimpanan, dan Akses Identitas dan Manajemen di luar kotak.

Amazon EKS Auto Mode menyertakan driver dan plugin perangkat yang diperlukan untuk jenis instans tertentu, seperti driver NVIDIA dan AWS Neuron. Anda tidak perlu menginstal atau memperbarui komponen ini.

Mode Otomatis EKS secara otomatis mengelola driver untuk akselerator ini:

- [AWS Trainium](#)
- [AWS Inferensia](#)
- [NVIDIA GPUs di Amazon EC2 mempercepat instans](#)

 Note

Mode Otomatis EKS menyertakan plugin perangkat NVIDIA untuk Kubernetes. Plugin ini berjalan secara otomatis dan tidak terlihat sebagai set daemon di cluster Anda.

Dukungan jaringan tambahan:

- [Adaptor Kain Elastis \(EFA\)](#)

Amazon EKS Auto Mode menghilangkan kerja keras driver akselerator dan manajemen plugin perangkat.

Anda juga bisa mendapatkan keuntungan dari penghematan biaya dengan menskalakan cluster ke nol. Anda dapat mengonfigurasi Mode Otomatis EKS untuk menghentikan instance saat tidak ada beban kerja yang berjalan. Ini berguna untuk beban kerja inferensi berbasis batch.

Berikut ini memberikan contoh cara meluncurkan beban kerja yang dipercepat dengan Amazon EKS Auto Mode.

Prasyarat

- Cluster Kubernetes dengan Amazon EKS Auto Mode dikonfigurasi.
- Kelas default EKS Node seperti yang dibuat saat general-purpose atau system Managed Node Pools diaktifkan.

Langkah 1: Menyebarkan beban kerja GPU

Dalam contoh ini, Anda akan membuat beban kerja berbasis NVIDIA NodePool untuk yang membutuhkan memori GPU 45GB. Dengan Mode Otomatis EKS, Anda menggunakan batasan penjadwalan Kubernetes untuk menentukan persyaratan instans Anda.

Untuk menerapkan Amazon EKS Auto Mode NodePool dan sampel workload, tinjau definisi berikut NodePool dan Pod dan simpan sebagai `nodepool-gpu.yaml` dan `pod.yaml`:

nodepool-gpu.yaml

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: gpu
spec:
  disruption:
    budgets:
      - nodes: 10%
    consolidateAfter: 1h
    consolidationPolicy: WhenEmpty
  template:
    metadata: {}
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default
      requirements:
        - key: "karpenter.sh/capacity-type"
          operator: In
          values: ["on-demand"]
        - key: "kubernetes.io/arch"
          operator: In
          values: ["amd64"]
        - key: "eks.amazonaws.com/instance-family"
          operator: In
          values:
            - g6e
            - g6
      taints:
        - key: nvidia.com/gpu
          effect: NoSchedule
      terminationGracePeriod: 24h0m0s
```

pod.yaml

```
apiVersion: v1
kind: Pod
```

```
metadata:
  name: nvidia-smi
spec:
  nodeSelector:
    eks.amazonaws.com/instance-gpu-name: l40s
    eks.amazonaws.com/compute-type: auto
  restartPolicy: OnFailure
  containers:
    - name: nvidia-smi
      image: public.ecr.aws/amazonlinux/amazonlinux:2023-minimal
      args:
        - "nvidia-smi"
      resources:
        requests:
          memory: "30Gi"
          cpu: "3500m"
          nvidia.com/gpu: 1
        limits:
          memory: "30Gi"
          nvidia.com/gpu: 1
  tolerations:
    - key: nvidia.com/gpu
      effect: NoSchedule
      operator: Exists
```

Perhatikan bahwa `eks.amazonaws.com/compute-type: auto` memilih mengharuskan beban kerja diterapkan pada node Mode Otomatis Amazon EKS. Ini NodePool juga menetapkan taint yang hanya memungkinkan pod dengan toleransi untuk Nvidia GPUs dijadwalkan.

Terapkan NodePool dan beban kerja ke cluster Anda.

```
kubectl apply -f nodepool-gpu.yaml
kubectl apply -f pod.yaml
```

Anda akan melihat output berikut:

```
nodepool.karpenter.sh/gpu configured created
pod/nvidia-smi created
```

Tunggu beberapa detik, dan periksa node di cluster Anda. Anda sekarang akan melihat node baru yang disediakan di kluster Mode Otomatis Amazon EKS Anda:


```
> kubectl get nodes
```

NAME	TYPE	CAPACITY	ZONE	NODE	READY	AGE
gpu-dnknr	g6e.2xlarge	on-demand	us-west-2b	i-02315c7d7643cdee6	True	76s

Langkah 2: Validasi

Anda dapat melihat Amazon EKS Auto Mode meluncurkan a, g6e.2xlarge bukan g6.2xlarge karena beban kerja memerlukan instance dengan l40sGPU, menurut batasan penjadwalan Kubernetes berikut:

```
...
nodeSelector:
  eks.amazonaws.com/instance-gpu-name: l40s
...
requests:
  memory: "30Gi"
  cpu: "3500m"
  nvidia.com/gpu: 1
limits:
  memory: "30Gi"
  nvidia.com/gpu: 1
```

Sekarang, lihat log kontainer, dengan menjalankan perintah berikut:

```
kubectl logs nvidia-smi
```

Contoh output:

```
+-----+
+
| NVIDIA-SMI 535.230.02                Driver Version: 535.230.02   CUDA Version: 12.2
|
|-----+-----|
+-----+
| GPU   Name                               Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC
|
| Fan   Temp   Perf              Pwr:Usage/Cap |      Memory-Usage | GPU-Util  Compute M.
|
|                               |                      |              MIG M.
|
```

```

|=====+=====
+=====|
|  0  NVIDIA L40S                      On | 00000000:30:00.0 Off | 0
|
| N/A    27C    P8                      23W / 350W | 0MiB / 46068MiB | 0%    Default
|
|                      |                      | N/A
|
+-----+-----
+-----+

+-----+
+
| Processes:
|
| GPU    GI    CI          PID    Type    Process name                      GPU Memory
|
|        ID    ID
|
|
=====|
| No running processes found
|
+-----+
+

```

Anda dapat melihat bahwa penampung telah mendeteksi itu berjalan pada instance dengan NVIDIA GPU dan Anda tidak perlu menginstal driver perangkat apa pun, karena ini dikelola oleh Amazon EKS Auto Mode.

Langkah 3: Membersihkan

Untuk menghapus semua objek yang dibuat, gunakan `kubectl` untuk menghapus penyebaran sampel NodePool sehingga node dihentikan:

```

kubectl delete -f nodepool-gpu.yaml
kubectl delete -f pod.yaml

```

Contoh NodePools Referensi

Buat NVIDIA NodePool

Berikut ini NodePool mendefinisikan:

- Hanya meluncurkan contoh g6e dan keluarga g6
- Konsolidasikan node saat kosong selama 1 jam
 - Nilai 1 jam untuk `consolidateAfter` mendukung beban kerja runcing dan mengurangi churn node. Anda dapat menyetel `consolidateAfter` berdasarkan persyaratan beban kerja Anda.

Contoh NodePool dengan keluarga instans GPU dan konsolidasi

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: gpu
spec:
  disruption:
    budgets:
      - nodes: 10%
    consolidateAfter: 1h
    consolidationPolicy: WhenEmpty
  template:
    metadata: {}
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default
      requirements:
        - key: "karpenter.sh/capacity-type"
          operator: In
          values: ["on-demand"]
        - key: "kubernetes.io/arch"
          operator: In
          values: ["amd64"]
        - key: "eks.amazonaws.com/instance-family"
          operator: In
          values:
            - g6e
            - g6
    terminationGracePeriod: 24h0m0s
```

Alih-alih menyetel yang mungkin `eks.amazonaws.com/instance-gpu-name` Anda gunakan `eks.amazonaws.com/instance-family` untuk menentukan keluarga instance. Untuk label

terkenal lainnya yang memengaruhi tinjauan penjadwalan, lihat [the section called “Label yang Didukung Mode Otomatis EKS”](#).

Jika Anda memiliki persyaratan penyimpanan khusus, Anda dapat menyetel penyimpanan singkat `nodeiops`, `size` dan `throughput` dengan membuat sendiri [NodeClass](#) untuk referensi di `NodePool`. Pelajari lebih lanjut tentang [NodeClass opsi yang dapat dikonfigurasi](#).

Contoh konfigurasi penyimpanan untuk NodeClass

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: gpu
spec:
  ephemeralStorage:
    iops: 3000
    size: 80Gi
    throughput: 125
```

Tentukan AWS Trainium dan AWS Inferensia NodePool

Berikut ini `NodePool` memiliki `eks.amazonaws.com/instance-category` satu set yang mengatakan, hanya meluncurkan contoh keluarga Inferentia dan Trainium:

```
- key: "eks.amazonaws.com/instance-category"
  operator: In
  values:
    - inf
    - trn
```

Buat laporan kepatuhan CIS dari node Kubernetes menggunakan `debug kubectl`

Topik ini menjelaskan cara membuat laporan kepatuhan CIS (Center for Internet Security) untuk node Amazon EKS menggunakan `kubectl debug` perintah. Perintah ini memungkinkan Anda untuk membuat kontainer debugging sementara pada node Kubernetes dan menjalankan pemeriksaan kepatuhan CIS menggunakan alat ini. `apiclient` Alat ini adalah bagian dari Bottlerocket OS, OS yang digunakan oleh node Mode Otomatis EKS.

Prasyarat

Sebelum Anda mulai, pastikan Anda memiliki:

- Akses ke kluster Amazon EKS dengan `kubectl` dikonfigurasi (versi minimal harus v1.32.0; ketik `kubectl version` untuk memeriksa).
- Izin IAM yang sesuai untuk men-debug node.
- Profil valid yang memungkinkan operasi debug (misalnya, `sysadmin`).

Untuk informasi selengkapnya tentang menggunakan profil debugging dengan `kubectl`, lihat [Debugging Pod atau Node saat menerapkan profil dalam dokumentasi](#) Kubernetes.

Prosedur

1. Tentukan ID AWS Instance dari node yang ingin Anda jalankan laporannya. Gunakan perintah berikut untuk daftar node dalam cluster. ID instance ditemukan di kolom nama, dan dimulai dengan `i-`:

```
kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
i-0ea0ba0f8ef9ad609	Ready	<none>	62s	v1.30.10-eks-1a9dacd

2. Jalankan perintah berikut, ganti `<instance-id>` dengan ID instance dari node yang ingin Anda kueri:

```
kubectl debug node/<instance-id> -it --profile=sysadmin --image=public.ecr.aws/amazonlinux/amazonlinux:2023 -- bash -c "yum install -q -y util-linux-core; nsenter -t 1 -m apiclient report cis --level 1 --format text"
```

Komponen dari perintah ini meliputi:

- `kubectl debug node/<instance-id>`— Membuat sesi debugging pada ID EC2 instance yang ditentukan.
- `-it`— Mengalokasikan TTY (shell baris perintah) dan membuat stdin tetap terbuka untuk penggunaan interaktif.
- `--profile=sysadmin`— Menggunakan `kubectl` profil yang ditentukan dengan izin yang sesuai.

- `--image=public.ecr.aws/amazonlinux/amazonlinux:2023`- Digunakan `amazonlinux:2023` sebagai gambar kontainer untuk debugging.
- `bash -c "..."` — Mengeksekusi perintah berikut dalam bash shell:
 - `yum install -q -y util-linux-core`— Diam-diam menginstal paket utilitas yang diperlukan.
 - `nsenter -t 1 -m`— Berjalan `nsenter` untuk masuk ke namespace dari proses host (PID 1).
 - `apiclient report cis --level 1 --format text`— Menjalankan laporan kepatuhan CIS di level 1 dengan output teks.

3. Tinjau keluaran teks laporan.

Menafsirkan output

Perintah tersebut menghasilkan laporan berbasis teks yang menunjukkan status kepatuhan berbagai kontrol CIS. Outputnya meliputi:

- Kontrol CIS individu IDs
- Deskripsi setiap kontrol
- Status Lulus, Gagal, atau Lewati untuk setiap pemeriksaan
- Detail yang menjelaskan masalah kepatuhan

Berikut adalah contoh output dari laporan yang dijalankan pada instance Bottlerocket:

```
Benchmark name:  CIS Bottlerocket Benchmark
Version:         v1.0.0
Reference:       https://www.cisecurity.org/benchmark/bottlerocket
Benchmark level: 1
Start time:      2025-04-11T01:40:39.055623436Z

[SKIP] 1.2.1      Ensure software update repositories are configured (Manual)
[PASS] 1.3.1      Ensure dm-verity is configured (Automatic)[PASS] 1.4.1      Ensure
setuid programs do not create core dumps (Automatic)
[PASS] 1.4.2      Ensure address space layout randomization (ASLR) is enabled
(Automatic)
[PASS] 1.4.3      Ensure unprivileged eBPF is disabled (Automatic)
[PASS] 1.5.1      Ensure SELinux is configured (Automatic)
```

```

[SKIP] 1.6      Ensure updates, patches, and additional security software are
                installed (Manual)
[PASS] 2.1.1.1  Ensure chrony is configured (Automatic)
[PASS] 3.2.5    Ensure broadcast ICMP requests are ignored (Automatic)
[PASS] 3.2.6    Ensure bogus ICMP responses are ignored (Automatic)
[PASS] 3.2.7    Ensure TCP SYN Cookies is enabled (Automatic)
[SKIP] 3.4.1.3  Ensure IPv4 outbound and established connections are configured
                (Manual)
[SKIP] 3.4.2.3  Ensure IPv6 outbound and established connections are configured
                (Manual)
[PASS] 4.1.1.1  Ensure journald is configured to write logs to persistent disk
                (Automatic)
[PASS] 4.1.2    Ensure permissions on journal files are configured (Automatic)

Passed:        11
Failed:        0
Skipped:       4
Total checks:  15

```

Untuk informasi tentang benchmark, lihat [Kubernetes Benchmark](#) dari Center for Internet Security (CIS).

Sumber daya terkait

- [Tolok Ukur CIS Bottlerocket](#) dalam Dokumentasi OS Bottlerocket.
- [Debug Running Pod dalam Dokumentasi](#) Kubernetes.
- [Benchmark Kubernetes](#) dari Pusat Keamanan Internet (CIS)

Aktifkan Enkripsi Volume EBS dengan Kunci KMS yang Dikelola Pelanggan untuk Mode Otomatis EKS

Anda dapat mengenkripsi volume root sementara untuk instans Mode Otomatis EKS dengan kunci KMS yang dikelola pelanggan.

Amazon EKS Auto Mode menggunakan peran terkait layanan untuk mendelegasikan izin ke AWS layanan lain saat mengelola volume EBS terenkripsi untuk kluster Kubernetes Anda. Topik ini menjelaskan cara menyiapkan kebijakan kunci yang Anda perlukan saat menentukan kunci terkelola pelanggan untuk enkripsi Amazon EBS dengan Mode Otomatis EKS.

Pertimbangan:

- Mode Otomatis EKS tidak memerlukan otorisasi tambahan untuk menggunakan kunci AWS terkelola default untuk melindungi volume terenkripsi di akun Anda.
- Topik ini mencakup mengenkripsi volume data, volume root untuk instance. EC2 Untuk informasi selengkapnya tentang mengenkripsi volume data yang digunakan untuk beban kerja, lihat. [the section called “Buat StorageClass”](#)

Gambaran Umum

Kunci AWS KMS berikut dapat digunakan untuk enkripsi volume root Amazon EBS saat Mode Otomatis EKS meluncurkan instance:

- AWS kunci terkelola — Kunci enkripsi di akun Anda yang dibuat, dimiliki, dan dikelola Amazon EBS. Ini adalah kunci enkripsi default untuk akun baru.
- Kunci terkelola pelanggan — Kunci enkripsi khusus yang Anda buat, miliki, dan kelola.

Note

Kuncinya harus simetris. Amazon EBS tidak mendukung kunci yang dikelola pelanggan asimetris.

Langkah 1: Konfigurasi kebijakan kunci

Kunci KMS Anda harus memiliki kebijakan kunci yang memungkinkan Mode Otomatis EKS meluncurkan instans dengan volume Amazon EBS yang dienkripsi dengan kunci yang dikelola pelanggan.

Konfigurasi kebijakan kunci Anda dengan struktur berikut:

Note

Kebijakan ini hanya mencakup izin untuk Mode Otomatis EKS. Kebijakan kunci mungkin memerlukan izin tambahan jika identitas lain perlu menggunakan kunci atau mengelola hibah.

```
{  
  "Version": "2012-10-17",  
  "Id": "MyKeyPolicy",
```



```

    "Statement": [
      {
        "Sid": "Allow use of the key",
        "Effect": "Allow",
        "Principal": {
          "AWS": [
            "arn:aws: iam::<account-id>:role/ClusterServiceRole"
          ]
        },
        "Action": [
          "kms:Encrypt",
          "kms:Decrypt",
          "kms:ReEncrypt*",
          "kms:GenerateDataKey*",
          "kms:DescribeKey"
        ],
        "Resource": "*"
      },
      {
        "Sid": "Allow attachment of persistent resources",
        "Effect": "Allow",
        "Principal": {
          "AWS": [
            "arn:aws: iam::<account-id>:role/ClusterServiceRole"
          ]
        },
        "Action": [
          "kms:CreateGrant",
          "kms:ListGrants",
          "kms:RevokeGrant"
        ],
        "Resource": "*",
        "Condition": {
          "Bool": {
            "kms:GrantIsForAWSResource": "true"
          }
        }
      }
    ]
  }
}

```

Pastikan untuk mengganti <account-id> dengan ID AWS akun Anda yang sebenarnya.

Saat mengonfigurasi kebijakan kunci:

- `ClusterServiceRole` harus memiliki izin IAM yang diperlukan untuk menggunakan kunci KMS untuk operasi enkripsi
- `kms:GrantIsForAWSResource` ini memastikan bahwa hibah hanya dapat dibuat untuk layanan AWS

Langkah 2: Konfigurasi NodeClass dengan kunci yang dikelola pelanggan Anda

Setelah mengonfigurasi kebijakan kunci, rujuk kunci KMS dalam konfigurasi Mode NodeClass Otomatis EKS Anda:

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: my-node-class
spec:
  # Insert existing configuration

  ephemeralStorage:
    size: "80Gi" # Range: 1-59000Gi or 1-64000G or 1-58Ti or 1-64T
    iops: 3000 # Range: 3000-16000
    throughput: 125 # Range: 125-1000

    # KMS key for encryption
    kmsKeyID: "arn:aws:kms:<region>:<account-id>:key/<key-id>"
```

Ganti nilai placeholder dengan nilai aktual Anda:

- `<region>` dengan AWS wilayah Anda
- `<account-id>` dengan ID AWS akun Anda
- `<key-id>` dengan ID kunci KMS Anda

Anda dapat menentukan kunci KMS menggunakan salah satu format berikut:

- ID Kunci KMS: `1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d`
- ARN Kunci KMS: `arn:aws:kms:us-west-2:111122223333:key/1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d`
- Nama Alias Kunci: `alias/eks-auto-mode-key`
- Kunci Alias ARN: `arn:aws:kms:us-west-2:111122223333:alias/eks-auto-mode-key`

Terapkan NodeClass konfigurasi menggunakan kubectl:

```
kubectl apply -f nodeclass.yaml
```

Sumber Daya Terkait

- [Buat Kelas Node untuk Amazon EKS](#)
- Lihat informasi selengkapnya di Panduan Pengembang Layanan Manajemen AWS Utama
 - [Izin untuk AWS layanan dalam kebijakan utama](#)
 - [Mengubah kebijakan utama](#)
 - [Hibah di AWS KMS](#)

Perbarui kontrol organisasi untuk Mode Otomatis EKS

Beberapa kontrol organisasi dapat mencegah Mode Otomatis EKS berfungsi dengan benar. Jika demikian, Anda harus memperbarui kontrol ini untuk memungkinkan Mode Otomatis EKS memiliki izin yang diperlukan untuk mengelola EC2 instance atas nama Anda.

Mode Otomatis EKS menggunakan peran layanan untuk meluncurkan EC2 Instans yang mendukung Node Mode Otomatis EKS. Peran layanan adalah peran [IAM](#) yang dibuat di akun Anda yang diasumsikan oleh layanan untuk melakukan tindakan atas nama Anda. [Kebijakan Kontrol Layanan](#) (SCPs) selalu berlaku untuk tindakan yang dilakukan dengan peran layanan. Hal ini memungkinkan SCP untuk menghambat operasi Mode Otomatis. Kejadian yang paling umum adalah ketika SCP digunakan untuk membatasi Amazon Machine Images (AMIs) yang dapat diluncurkan. Untuk memungkinkan Mode Otomatis EKS berfungsi, ubah SCP untuk mengizinkan peluncuran AMIs dari akun Mode Otomatis EKS.

Anda juga dapat menggunakan AMIs fitur [EC2 Diizinkan](#) untuk membatasi visibilitas AMIs di akun lain. Jika Anda menggunakan fitur ini, Anda harus memperluas kriteria gambar untuk juga menyertakan akun AMI Mode Otomatis EKS di wilayah yang diminati.

Contoh SCP untuk memblokir semua AMIs kecuali untuk Mode Otomatis EKS AMIs

SCP di bawah ini mencegah panggilan `ec2:RunInstances` kecuali AMI milik akun AMI Mode Otomatis EKS untuk `us-west-2` atau `us-east-1`.

Note

Penting untuk tidak menggunakan kunci `ec2:Owner` konteks. Amazon memiliki akun AMI Mode Otomatis EKS dan nilai untuk kunci ini akan `amazon` selalu ada. Membangun SCP yang memungkinkan peluncuran AMIs jika `ec2:Owner` is `amazon` akan memungkinkan peluncuran AMI milik Amazon, bukan hanya untuk Mode Otomatis EKS. *

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyAMI",
      "Effect": "Deny",
      "Action": "ec2:RunInstances",
      "Resource": "arn:*:ec2:*:image/ami-*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceAccount": [
            "767397842682",
            "992382739861"
          ]
        }
      }
    }
  ]
}
```

Akun AMI Mode Otomatis EKS

AWS akun yang bervariasi menurut wilayah host EKS Auto Mode publik AMIs.

AWS Wilayah	Akun
af-south-1	471112993317
ap-east-1	590183728416
ap-northeast-1	851725346105

ap-northeast-2	992382805010
ap-northeast-3	891377407544
ap-south-1	975049899075
ap-south-2	590183737426
ap-southeast-1	339712723301
ap-southeast-2	58264376476
ap-southeast-3	471112941769
ap-southeast-4	590183863144
ap-southeast-5	654654202513
ap-tenggara 7	533267217478
ca-central-1	992382439851
ca-west-1	767397959864
eu-central-1	891376953411
eu-central-2	381492036002
eu-north-1	339712696471
eu-south-1	975049955519
eu-south-2	471112620929
eu-west-1	381492008532
eu-west-2	590184142468
eu-west-3	891376969258
il-central-1	590183797093

me-central-1	637423494195
me-south-1	905418070398
mx-pusat-1	211125506622
sa-east-1	339712709251
us-east-1	992382739861
us-east-2	975050179949
us-west-1	975050035094
us-west-2	767397842682

Kontrol penyebaran beban kerja ke Reservasi Kapasitas EC2 Sesuai Permintaan dengan Mode Otomatis EKS

EC2 Reservasi Kapasitas Sesuai Permintaan (ODCRs) memungkinkan Anda memesan kapasitas komputasi untuk EC2 instans Amazon Anda di Availability Zone tertentu untuk durasi berapa pun. Saat menggunakan Mode Otomatis EKS, Anda mungkin ingin mengontrol apakah beban kerja Kubernetes Anda diterapkan ke instans cadangan ini untuk memaksimalkan pemanfaatan kapasitas yang telah dibeli sebelumnya atau untuk memastikan beban kerja kritis memiliki akses ke sumber daya yang terjamin.

Secara default, Mode Otomatis EKS secara otomatis diluncurkan ke terbuka ODCRs. Namun, dengan mengonfigurasi `capacityReservationSelectorTerms` pada a `NodeClass`, Anda dapat secara eksplisit mengontrol beban kerja yang ODCRs Anda gunakan. Node yang disediakan menggunakan konfigurasi ODCRs akan memiliki `karpenter.sh/capacity-type: reserved` dan akan diprioritaskan daripada on-demand dan spot. Setelah fitur ini diaktifkan, Mode Otomatis EKS tidak akan lagi secara otomatis menggunakan open ODCRs —mereka harus dipilih secara eksplisit oleh a `NodeClass`, memberi Anda kontrol yang tepat atas penggunaan reservasi kapasitas di seluruh cluster Anda.

⚠ Warning

Jika Anda mengonfigurasi `capacityReservationSelectorTerms` di `NodeClass` dalam cluster, Mode Otomatis EKS tidak akan lagi secara otomatis menggunakan open ODCRs untuk apa pun `NodeClass` di cluster.

Contoh NodeClass

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
spec:
  # Optional: Selects upon on-demand capacity reservations and capacity blocks
  # for EKS Auto Mode to prioritize.
  capacityReservationSelectorTerms:
    - id: cr-56fac701cc1951b03
    # Alternative Approaches
    - tags:
        app: "my-app"
    # Optional owning account ID filter
    owner: "012345678901"
```

Contoh ini `NodeClass` menunjukkan dua pendekatan untuk memilih ODCRs. Metode pertama secara langsung mereferensikan ODCR tertentu dengan ID (`cr-56fac701cc1951b03`). Metode kedua menggunakan pemilihan berbasis tag, menargetkan ODCRs dengan tag. Name: `"targeted-odcr"` Anda juga dapat memfilter secara opsional berdasarkan AWS akun yang memiliki reservasi, yang sangat berguna dalam skenario lintas akun atau saat bekerja dengan reservasi kapasitas bersama.

Pelajari cara kerja Mode Otomatis EKS

Gunakan chapter ini untuk mempelajari cara kerja komponen cluster Amazon EKS Auto Mode.

Topik

- [Pelajari tentang instans Terkelola Mode Otomatis Amazon EKS](#)
- [Pelajari tentang identitas dan akses dalam Mode Otomatis EKS](#)
- [Pelajari tentang Jaringan VPC dan Load Balancing dalam Mode Otomatis EKS](#)

Pelajari tentang instans Terkelola Mode Otomatis Amazon EKS

Topik ini menjelaskan cara Amazon EKS Auto Mode mengelola EC2 instans Amazon di kluster EKS Anda. Saat Anda mengaktifkan Mode Otomatis EKS, sumber daya komputasi kluster Anda secara otomatis disediakan dan dikelola oleh EKS, mengubah cara Anda berinteraksi dengan EC2 instance yang berfungsi sebagai node di cluster Anda.

Memahami cara Amazon EKS Auto Mode mengelola instans sangat penting untuk merencanakan strategi penerapan beban kerja dan prosedur operasional Anda. Tidak seperti EC2 instance tradisional atau grup node terkelola, instance ini mengikuti model siklus hidup yang berbeda di mana EKS bertanggung jawab atas banyak aspek operasional, sambil membatasi jenis akses dan penyesuaian tertentu.

Amazon EKS Auto Mode mengotomatiskan tugas rutin untuk membuat EC2 Instans baru, dan melampirkannya sebagai node ke cluster EKS Anda. Mode Otomatis EKS mendeteksi ketika beban kerja tidak dapat masuk ke node yang ada, dan membuat Instance baru EC2 .

Amazon EKS Auto Mode bertanggung jawab untuk membuat, menghapus, dan menambal Instans EC2 . Anda bertanggung jawab atas kontainer dan pod yang digunakan pada instance.

EC2 Instans yang dibuat oleh EKS Auto Mode berbeda dari EC2 Instans lain, mereka adalah instance terkelola. Instans terkelola ini dimiliki oleh EKS dan lebih dibatasi. Anda tidak dapat langsung mengakses atau menginstal perangkat lunak pada instans yang dikelola oleh Mode Otomatis EKS.

AWS menyarankan menjalankan Mode Otomatis EKS atau Karpenter yang dikelola sendiri. Anda dapat menginstal keduanya selama migrasi atau dalam konfigurasi lanjutan. Jika Anda telah menginstal keduanya, konfigurasi kumpulan node Anda sehingga beban kerja dikaitkan dengan Karpenter atau Mode Otomatis EKS.

Untuk informasi selengkapnya, lihat [instans EC2 terkelola Amazon](#) di panduan EC2 pengguna Amazon.

Tabel perbandingan

EC2 Instans Standar	Contoh terkelola Mode Otomatis EKS
Anda bertanggung jawab untuk menambal dan memperbarui instance.	AWS secara otomatis menambal dan memperbarui instance.

EC2 Instans Standar	Contoh terkelola Mode Otomatis EKS
EKS tidak bertanggung jawab atas perangkat lunak pada instance.	EKS bertanggung jawab atas perangkat lunak tertentu pada instance, seperti <code>kubelet</code> , runtime kontainer, dan sistem operasi.
Anda dapat menghapus EC2 Instance menggunakan EC2 API.	EKS menentukan jumlah instans yang digunakan di akun Anda. Jika Anda menghapus beban kerja, EKS akan mengurangi jumlah instance di akun Anda.
Anda dapat menggunakan SSH untuk mengakses EC2 Instance.	Anda dapat menerapkan pod dan kontainer ke instance terkelola.
Anda menentukan sistem operasi dan gambar (AMI).	AWS menentukan sistem operasi dan gambar.
Anda dapat menerapkan beban kerja yang mengandalkan fungsionalitas Windows atau Ubuntu.	Anda dapat menyebarkan kontainer berbasis Linux, tetapi tanpa dependensi OS tertentu.
Anda menentukan jenis instance dan keluarga apa yang akan diluncurkan.	AWS menentukan jenis instance dan keluarga apa yang akan diluncurkan. Anda dapat menggunakan Node Pool untuk membatasi jenis instance EKS Auto Mode memilih dari.

Fungsionalitas berikut berfungsi untuk instans Terkelola dan instans Standar EC2 :

- Anda dapat melihat instance di AWS konsol.
- Anda dapat menggunakan penyimpanan instance sebagai penyimpanan sementara untuk beban kerja.

AMI Support

Dengan Mode Otomatis EKS, AWS tentukan image (AMI) yang digunakan untuk node komputasi Anda. AWS memantau peluncuran versi AMI Mode Otomatis EKS baru. Jika Anda mengalami masalah beban kerja yang terkait dengan versi AMI, buat kasus dukungan. Untuk informasi

selengkapnya, lihat [Membuat kasus dukungan dan manajemen kasus](#) di Panduan Pengguna AWS Support.

Umumnya, EKS merilis AMI baru setiap minggu yang berisi CVE dan perbaikan keamanan.

Referensi instans yang didukung Mode Otomatis EKS

Mode Otomatis EKS hanya membuat instance tipe yang didukung, dan memenuhi persyaratan ukuran minimum.

Mode Otomatis EKS mendukung jenis contoh berikut:

Rangkaian	Tipe instans
Komputasi Dioptimalkan (C)	c8g, c7a, c7g, c7gn, c7gd, c7i, c7i-flex, c6a, c6g, c6i, c6gn, c6id, c6in, c6gd, c5, c5a, c5d, c5ad, c5n, c4
Tujuan Umum (M)	m8g, m7i, m7a, m7g, m7gd, m7i-flex, m6a, m6i, m6in, m6g, m6idn, m6id, m6gd, m5, m5a, m5ad, m5n, m5dn, m5d, m5zn, m4
Memori Dioptimalkan (R)	r8g, r7a, r7iz, r7gd, r7i, r7g, r6a, r6i, r6id, r6in, r6idn, r6g, r6gd, r5, r5n, r5a, r5dn, r5b, r5ad, r5d, r4
Burstable (T)	t4g, t3, t3a, t2
Memori Tinggi (Z/X)	z1d, x8g, x2gd
Penyimpanan Dioptimalkan (I/ D)	i4g, i4i, i3, i3en, is4gen, d3, d3en, im4gn
Komputasi yang Dipercepat (P/G/ Inf/Trn)	p5, p4d, p3, p3dn, gr6, g6, g6e, g5g, g5, g4dn, inf2, inf1, trn1, trn1n
Komputasi Kinerja Tinggi (X2)	x2iezn, x2iedn, x2idn

Selain itu, Mode Otomatis EKS hanya akan membuat EC2 instance yang memenuhi persyaratan berikut:

- Lebih dari 1 CPU
- Ukuran instans tidak nano, mikro atau kecil

Untuk informasi selengkapnya, lihat [konvensi penamaan jenis EC2 instans Amazon](#).

Layanan Metadata Instans

- Mode Otomatis EKS diberlakukan IMDSv2 dengan batas lompatan 1 secara default, mengikuti praktik terbaik AWS keamanan.
- Konfigurasi default ini tidak dapat diubah dalam Mode Otomatis.
- Untuk add-on yang biasanya memerlukan akses IMDS, berikan parameter (seperti AWS wilayah) selama instalasi untuk menghindari pencarian IMDS. Untuk informasi selengkapnya, lihat [the section called “Bidang yang dapat Anda sesuaikan”](#).
- Jika sebuah Pod benar-benar membutuhkan akses IMDS ketika berjalan dalam Mode Otomatis, Pod harus dikonfigurasi untuk dijalankan. `hostNetwork: true` Hal ini memungkinkan Pod untuk mengakses layanan metadata instance secara langsung.
- Pertimbangkan implikasi keamanan saat memberikan akses Pod ke metadata instance.

Untuk informasi selengkapnya tentang Amazon EC2 Instance Metadata Service (IMDS), lihat [Mengonfigurasi opsi Layanan Metadata Instans di Panduan Pengguna Amazon. EC2](#)

Pertimbangan

- Jika penyimpanan sementara yang dikonfigurasi di dalam NodeClass lebih kecil dari penyimpanan NVMe lokal untuk instance, Mode Otomatis EKS menghilangkan kebutuhan untuk konfigurasi manual dengan secara otomatis mengambil tindakan berikut:
 - Menggunakan volume data Amazon EBS yang lebih kecil (20 GiB) untuk mengurangi biaya.
 - Memformat dan mengkonfigurasi penyimpanan NVMe lokal untuk penggunaan data sementara. Ini termasuk menyiapkan array RAID 0 jika ada beberapa NVMe drive.
- Amazon EKS Auto Mode tidak mendukung Layanan Injeksi AWS Kesalahan. Untuk informasi selengkapnya, lihat [Mengelola eksperimen Layanan Injeksi Kesalahan](#) di Panduan Pengguna AWS Resilience Hub.

- Anda tidak perlu menginstal node Mode Otomatis EKS. Neuron Device Plugin

Jika Anda memiliki jenis node lain di cluster Anda, Anda perlu mengonfigurasi plugin Perangkat Neuron agar tidak berjalan di node Mode Otomatis. Lihat informasi yang lebih lengkap di [the section called “Kontrol penyebaran”](#).

Pelajari tentang identitas dan akses dalam Mode Otomatis EKS

Topik ini menjelaskan peran dan izin Identity and Access Management (IAM) and Access Management (IAM) yang diperlukan untuk menggunakan Mode Otomatis EKS. Mode Otomatis EKS menggunakan dua peran IAM utama: Peran IAM Cluster dan Peran IAM Node. Peran ini bekerja sama dengan EKS Pod Identity dan entri akses EKS untuk menyediakan manajemen akses komprehensif untuk kluster EKS Anda.

Saat Anda mengonfigurasi Mode Otomatis EKS, Anda perlu mengatur peran IAM ini dengan izin khusus yang memungkinkan AWS layanan berinteraksi dengan sumber daya cluster Anda. Ini termasuk izin untuk mengelola sumber daya komputasi, volume penyimpanan, penyeimbang beban, dan komponen jaringan. Memahami konfigurasi peran ini sangat penting untuk operasi dan keamanan kluster yang tepat.

Dalam Mode Otomatis EKS, peran AWS IAM secara otomatis dipetakan ke izin Kubernetes melalui entri akses EKS, menghilangkan kebutuhan untuk konfigurasi manual atau binding kustom. `aws-auth` ConfigMaps Saat Anda membuat kluster mode otomatis baru, EKS secara otomatis membuat izin Kubernetes yang sesuai menggunakan entri Access, memastikan bahwa AWS layanan dan komponen kluster memiliki tingkat akses yang sesuai di dalam sistem otorisasi Kubernetes dan Kubernetes. AWS Integrasi otomatis ini mengurangi kompleksitas konfigurasi dan membantu mencegah masalah terkait izin yang biasanya terjadi saat mengelola kluster EKS.

IAM role kluster

Peran IAM Cluster adalah peran AWS Identity and Access Management (IAM) yang digunakan oleh Amazon EKS untuk mengelola izin kluster Kubernetes. Peran ini memberikan Amazon EKS izin yang diperlukan untuk berinteraksi dengan AWS layanan lain atas nama kluster Anda, dan secara otomatis dikonfigurasi dengan izin Kubernetes menggunakan entri akses EKS.

- Anda harus melampirkan kebijakan AWS IAM ke peran ini.
- Mode Otomatis EKS melampirkan izin Kubernetes ke peran ini secara otomatis menggunakan entri akses EKS.

- Dengan Mode Otomatis EKS, AWS sarankan untuk membuat Peran IAM Cluster tunggal per AWS akun.
- AWS menyarankan penamaan peran iniAmazonEKSAutoClusterRole.
- Peran ini memerlukan izin untuk beberapa AWS layanan untuk mengelola sumber daya termasuk volume EBS, Elastic Load Balancers, dan instance. EC2
- Konfigurasi yang disarankan untuk peran ini mencakup beberapa kebijakan IAM AWS terkelola, terkait dengan berbagai kemampuan Mode Otomatis EKS.
 - AmazonEKSCoordinatePolicy
 - AmazonEKSBlockStoragePolicy
 - AmazonEKSLoadBalancingPolicy
 - AmazonEKSNetworkingPolicy
 - AmazonEKSClusterPolicy

Untuk informasi selengkapnya tentang Peran IAM Cluster dan kebijakan IAM AWS terkelola, lihat:

- [the section called “ AWS kebijakan terkelola”](#)
- [the section called “IAM role klaster”](#)

Untuk informasi selengkapnya tentang akses Kubernetes, lihat:

- [the section called “Tinjau kebijakan akses”](#)

IAM role simpul

Peran Node IAM adalah peran AWS Identity and Access Management (IAM) yang digunakan oleh Amazon EKS untuk mengelola izin bagi node pekerja di klaster Kubernetes. Peran ini memberikan EC2 instans yang berjalan sebagai node Kubernetes izin yang diperlukan untuk berinteraksi dengan AWS layanan dan sumber daya, dan secara otomatis dikonfigurasi dengan izin Kubernetes RBAC menggunakan entri akses EKS.

- Anda harus melampirkan kebijakan AWS IAM ke peran ini.
- Mode Otomatis EKS melampirkan izin Kubernetes RBAC ke peran ini secara otomatis menggunakan entri akses EKS.
- AWS menyarankan penamaan peran iniAmazonEKSAutoNodeRole.

- Dengan Mode Otomatis EKS, AWS menyarankan untuk membuat Peran IAM Node tunggal per AWS akun.
- Peran ini memiliki izin terbatas. Izin utama termasuk mengasumsikan Peran Identitas Pod, dan menarik gambar dari ECR.
- AWS menyarankan kebijakan IAM AWS terkelola berikut:
 - `AmazonEKSEKSWorkerNodeMinimalPolicy`
 - `AmazonEC2ContainerRegistryPullOnly`

Untuk informasi selengkapnya tentang Peran IAM Cluster dan kebijakan IAM AWS terkelola, lihat:

- [the section called “ AWS kebijakan terkelola”](#)
- [the section called “IAM role simpul”](#)

Untuk informasi selengkapnya tentang akses Kubernetes, lihat:

- [the section called “Tinjau kebijakan akses”](#)

Peran terkait layanan

Amazon EKS menggunakan peran terkait layanan (SLR) untuk operasi tertentu. Peran terkait layanan adalah jenis IAM role unik yang terkait langsung dengan Amazon EKS. Peran terkait layanan telah ditentukan sebelumnya oleh Amazon EKS dan mencakup semua izin yang diperlukan layanan untuk memanggil AWS layanan lain atas nama Anda.

AWS secara otomatis membuat dan mengkonfigurasi SLR. Anda dapat menghapus SLR hanya setelah pertama kali menghapus sumber daya terkait mereka. Ini melindungi sumber daya Amazon EKS Anda karena Anda tidak dapat secara tidak sengaja menghapus izin untuk mengakses sumber daya.

Kebijakan SLR memberikan izin Amazon EKS untuk mengamati dan menghapus komponen infrastruktur inti: EC2 sumber daya (instance, antarmuka jaringan, grup keamanan), sumber daya ELB (penyeimbang beban, grup target), CloudWatch kemampuan (logging dan metrik), dan peran IAM dengan awalan “eks”. Ini juga memungkinkan jaringan titik akhir pribadi melalui asosiasi VPC/ hosted zona dan mencakup izin untuk EventBridge pemantauan dan pembersihan sumber daya yang ditandai EKS.

Untuk informasi selengkapnya, lihat:

- [the section called “AWS kebijakan terkelola: Amazon EKSService RolePolicy”](#)
- [the section called “Izin peran terkait layanan untuk Amazon EKS”](#)

AWS Tag khusus untuk sumber daya EKS Auto

Secara default, kebijakan terkelola yang terkait dengan Mode Otomatis EKS tidak mengizinkan penerapan tag yang ditentukan pengguna ke sumber daya yang disediakan AWS Mode Otomatis. Jika ingin menerapkan tag yang ditentukan pengguna ke AWS sumber daya, Anda harus melampirkan izin tambahan ke Peran IAM Cluster dengan izin yang cukup untuk membuat dan memodifikasi tag pada sumber daya. AWS Di bawah ini adalah contoh kebijakan yang memungkinkan akses penandaan tidak terbatas:

Lihat contoh kebijakan tag kustom

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Compute",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateFleet",
        "ec2:RunInstances",
        "ec2:CreateLaunchTemplate"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/eks:eks-cluster-name": "${aws:PrincipalTag/eks:eks-cluster-name}"
        },
        "StringLike": {
          "aws:RequestTag/eks:kubernetes-node-class-name": "*",
          "aws:RequestTag/eks:kubernetes-node-pool-name": "*"
        }
      }
    },
    {
      "Sid": "Storage",
      "Effect": "Allow",
      "Action": [
```

```

        "ec2:CreateVolume",
        "ec2:CreateSnapshot"
    ],
    "Resource": [
        "arn:aws: ec2:*:*:volume/*",
        "arn:aws: ec2:*:*:snapshot/*"
    ],
    "Condition": {
        "StringEquals": {
            "aws:RequestTag/eks:eks-cluster-name": "${aws:PrincipalTag/eks:eks-
cluster-name}"
        }
    }
},
{
    "Sid": "Networking",
    "Effect": "Allow",
    "Action": "ec2:CreateNetworkInterface",
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "aws:RequestTag/eks:eks-cluster-name": "${aws:PrincipalTag/eks:eks-
cluster-name}"
        },
        "StringLike": {
            "aws:RequestTag/eks:kubernetes-cni-node-name": "*"
        }
    }
},
{
    "Sid": "LoadBalancer",
    "Effect": "Allow",
    "Action": [
        "elasticloadbalancing:CreateLoadBalancer",
        "elasticloadbalancing:CreateTargetGroup",
        "elasticloadbalancing:CreateListener",
        "elasticloadbalancing:CreateRule",
        "ec2:CreateSecurityGroup"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "aws:RequestTag/eks:eks-cluster-name": "${aws:PrincipalTag/eks:eks-
cluster-name}"

```



```

        }
    },
    {
        "Sid": "ShieldProtection",
        "Effect": "Allow",
        "Action": [
            "shield:CreateProtection"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:RequestTag/eks:eks-cluster-name": "${aws:PrincipalTag/eks:eks-
cluster-name}"
            }
        }
    },
    {
        "Sid": "ShieldTagResource",
        "Effect": "Allow",
        "Action": [
            "shield:TagResource"
        ],
        "Resource": "arn:aws:shield:::protection/*",
        "Condition": {
            "StringEquals": {
                "aws:RequestTag/eks:eks-cluster-name": "${aws:PrincipalTag/eks:eks-
cluster-name}"
            }
        }
    }
]
}

```

Referensi Kebijakan Akses

Untuk informasi selengkapnya tentang izin Kubernetes yang digunakan oleh Mode Otomatis EKS, lihat [the section called “Tinjau kebijakan akses”](#)

Pelajari tentang Jaringan VPC dan Load Balancing dalam Mode Otomatis EKS

Topik ini menjelaskan cara mengonfigurasi jaringan Virtual Private Cloud (VPC) dan fitur load balancing di Mode Otomatis EKS. Sementara EKS Auto Mode mengelola sebagian besar komponen jaringan secara otomatis, Anda masih dapat menyesuaikan aspek-aspek tertentu dari konfigurasi jaringan cluster Anda melalui `NodeClass` sumber daya dan anotasi penyeimbang beban.

Saat Anda menggunakan Mode Otomatis EKS, AWS mengelola konfigurasi VPC Container Network Interface (CNI) dan penyediaan penyeimbang beban untuk klaster Anda. Anda dapat memengaruhi perilaku jaringan dengan mendefinisikan `NodeClass` objek dan menerapkan anotasi spesifik ke sumber daya Layanan dan Ingress Anda, sambil mempertahankan model operasional otomatis yang disediakan oleh Mode Otomatis EKS.

Kemampuan jaringan

EKS Auto Mode memiliki kemampuan jaringan baru yang menangani jaringan node dan pod. Anda dapat mengonfigurasinya dengan membuat objek `NodeClass` Kubernetes.

Opsi konfigurasi untuk AWS VPC CNI sebelumnya tidak akan berlaku untuk Mode Otomatis EKS.

Konfigurasikan jaringan dengan **NodeClass**

`NodeClass` Sumber daya dalam Mode Otomatis EKS memungkinkan Anda untuk menyesuaikan aspek-aspek tertentu dari kemampuan jaringan. Melalui `NodeClass`, Anda dapat menentukan pilihan grup keamanan, mengontrol penempatan node di seluruh subnet VPC, mengatur kebijakan SNAT, mengonfigurasi kebijakan jaringan, dan mengaktifkan pencatatan peristiwa jaringan. Pendekatan ini mempertahankan model operasional otomatis dari Mode Otomatis EKS sambil memberikan fleksibilitas untuk kustomisasi jaringan.

Anda dapat menggunakan a `NodeClass` untuk:

- Pilih Grup Keamanan untuk Node
- Kontrol bagaimana node ditempatkan pada Subnet VPC
- Mengatur Kebijakan SNAT Node ke `random` atau `disabled`
- Aktifkan kebijakan jaringan Kubernetes termasuk:
 - Setel Kebijakan Jaringan ke `Default Deny` atau `Default Allow`
 - Aktifkan Network Event Logging ke file.

- Mengisolasi lalu lintas pod dari lalu lintas node dengan melampirkan pod ke subnet yang berbeda.

Pelajari cara [Membuat Amazon EKS NodeClass](#).

Pertimbangan

Mode Otomatis EKS mendukung:

- Kebijakan Jaringan EKS.
- HostNetworkOpsi HostPort dan untuk Kubernetes Pods.
- Node dan Pod di subnet publik atau pribadi.
- Menyimpan kueri DNS pada node.

Mode Otomatis EKS tidak mendukung:

- Grup Keamanan per Pod (SGPP).
- Jaringan Kustom diENIConfig. Anda dapat menempatkan pod di beberapa subnet atau secara eksklusif mengisolasinya dari lalu lintas node dengan. [the section called “Pemilihan subnet untuk Pod”](#)
- IP hangat, awalan hangat, dan konfigurasi ENI hangat.
- Konfigurasi target IP minimum.
- Mengaktifkan atau menonaktifkan delegasi awalan.
- Konfigurasi lain yang didukung oleh AWS VPC CNI open source.
- Konfigurasi Kebijakan Jaringan seperti penyesuaian timer conntrack (default adalah 300s).
- Mengekspor log peristiwa jaringan ke CloudWatch.

Manajemen Sumber Daya Jaringan

Mode Otomatis EKS menangani awalan, pengalamatan IP, dan manajemen antarmuka jaringan dengan memantau NodeClass sumber daya untuk konfigurasi jaringan. Layanan ini melakukan beberapa operasi utama secara otomatis:

Delegasi Awalan

Mode Otomatis EKS /28 IPv4 menyediakan awalan ke antarmuka jaringan utama untuk node dan mempertahankan kumpulan sumber daya hangat yang telah ditentukan sebelumnya yang diskalakan

berdasarkan jumlah pod terjadwal. Bila perlu, ia menyediakan antarmuka jaringan sekunder dengan grup keamanan yang identik sebagai antarmuka utama di subnet node. Jika awalan tidak lagi tersedia di subnet, layanan akan kembali ke alamat sekunder. IPv4

Manajemen Cooldown

Layanan ini mengimplementasikan kumpulan cooldown untuk awalan atau IPv4 alamat sekunder yang tidak lagi digunakan. Setelah periode cooldown berakhir, sumber daya ini dilepaskan kembali ke VPC. Namun, jika pod menggunakan kembali sumber daya ini selama periode cooldown, mereka dipulihkan dari kolam cooldown.

IPv6 Support

Untuk IPv6 cluster, Mode Otomatis EKS menyediakan /80 IPv6 awalan per node pada antarmuka jaringan utama.

Layanan ini juga memastikan manajemen yang tepat dan pengumpulan sampah dari semua antarmuka jaringan.

Penyeimbangan beban

Anda mengonfigurasi AWS Elastic Load Balancer yang disediakan oleh Mode Otomatis EKS menggunakan anotasi pada resource Service dan Ingress.

Untuk informasi selengkapnya, lihat [the section called “Buat kelas ingress”](#) atau [the section called “Buat layanan”](#).

Pertimbangan untuk penyeimbangan beban dengan Mode Otomatis EKS

- Mode penargetan default adalah Mode IP, bukan Mode Instance.
- Mode Otomatis EKS hanya mendukung Mode Grup Keamanan untuk Network Load Balancer.
- AWS tidak mendukung migrasi penyeimbang beban dari pengontrol penyeimbang AWS beban yang dikelola sendiri ke manajemen dengan Mode Otomatis EKS.
- `networking.ingress.ipBlockBidang` dalam `TargetGroupBinding` spesifikasi tidak didukung.
- Jika node pekerja Anda menggunakan grup keamanan khusus (bukan pola `eks-cluster-sg` penamaan), peran klaster Anda memerlukan izin IAM tambahan. Kebijakan default yang dikelola EKS hanya mengizinkan EKS untuk memodifikasi grup keamanan bernama `eks-cluster-sg`.

Tanpa izin untuk memodifikasi grup keamanan kustom Anda, EKS tidak dapat menambahkan aturan masuk yang diperlukan yang memungkinkan lalu lintas ALB/NLB mencapai pod Anda.

Memecahkan Masalah Mode Otomatis EKS

Dengan Mode Otomatis EKS, AWS tanggung jawab lebih besar untuk EC2 Instans di akun Anda AWS . EKS bertanggung jawab atas runtime kontainer pada node, sistem operasi pada node, dan pengontrol tertentu. Ini termasuk pengontrol penyimpanan blok, pengontrol penyeimbang beban, dan pengontrol komputasi.

Anda harus menggunakan AWS dan Kubernetes APIs untuk memecahkan masalah node. Anda dapat:

- Gunakan `NodeDiagnostic` sumber daya Kubernetes untuk mengambil log node dengan menggunakan file. [the section called “Agen pemantauan simpul”](#) Untuk langkah lebih lanjut, lihat [the section called “Dapatkan log simpul”](#).
- Gunakan perintah AWS EC2 CLI `get-console-output` untuk mengambil output konsol dari node. Untuk langkah lebih lanjut, lihat [the section called “Dapatkan output konsol dari instance EC2 terkelola dengan menggunakan AWS EC2 CLI”](#).
- Gunakan kontainer debugging Kubernetes untuk mengambil log node. Untuk langkah lebih lanjut, lihat [the section called “Dapatkan log node dengan menggunakan wadah debug dan CLI kubectl”](#).

Note

Mode Otomatis EKS menggunakan instance EC2 terkelola. Anda tidak dapat mengakses instans EC2 terkelola secara langsung, termasuk oleh SSH.

Anda mungkin memiliki masalah berikut yang memiliki solusi khusus untuk komponen Mode Otomatis EKS:

- Pod terjebak dalam Pending status, yang tidak dijadwalkan ke node Mode Otomatis. Untuk solusi lihat [the section called “Memecahkan masalah Pod yang gagal menjadwalkan ke node Mode Otomatis”](#).
- EC2 instance terkelola yang tidak bergabung dengan cluster sebagai node Kubernetes. Untuk solusi lihat [the section called “Memecahkan masalah node yang tidak bergabung dengan cluster”](#).

- Kesalahan dan masalah dengan `NodePools`, `PersistentVolumes`, dan `Services` yang menggunakan pengontrol yang disertakan dalam Mode Otomatis EKS. Untuk solusi lihat [the section called “Memecahkan masalah pengontrol yang disertakan dalam Mode Otomatis”](#).
- Keamanan Pod yang ditingkatkan mencegah berbagi volume di seluruh Pod. Untuk solusi lihat [the section called “Berbagi Volume di Seluruh Pod”](#).

Anda dapat menggunakan metode berikut untuk memecahkan masalah komponen Mode Otomatis EKS:

- [the section called “Dapatkan output konsol dari instance EC2 terkelola dengan menggunakan AWS EC2 CLI”](#)
- [the section called “Dapatkan log node dengan menggunakan wadah debug dan CLI `kubectl`”](#)
- [the section called “Melihat sumber daya yang terkait dengan Mode Otomatis EKS di AWS Konsol”](#)
- [the section called “Lihat Masalah IAM di akun Anda AWS”](#)
- [the section called “Mendeteksi masalah konektivitas node dengan VPC Reachability Analyzer”](#)

Agen pemantauan simpul

Mode Otomatis EKS mencakup agen pemantauan simpul Amazon EKS. Anda dapat menggunakan agen ini untuk melihat pemecahan masalah dan debugging informasi tentang node. Agen pemantauan node menerbitkan Kubernetes events dan node. conditions Untuk informasi selengkapnya, lihat [the section called “Kesehatan simpul”](#).

Dapatkan output konsol dari instance EC2 terkelola dengan menggunakan AWS EC2 CLI

Prosedur ini membantu mengatasi masalah waktu boot atau masalah tingkat kernel.

Pertama, Anda perlu menentukan ID EC2 Instance dari instance yang terkait dengan beban kerja Anda. Kedua, gunakan AWS CLI untuk mengambil output konsol.

1. Konfirmasikan bahwa Anda telah `kubectl` menginstal dan terhubung ke cluster Anda
2. (Opsional) Gunakan nama Deployment Kubernetes untuk membuat daftar pod terkait.

```
kubectl get pods -l app=<deployment-name>
```

- Gunakan nama Pod Kubernetes untuk menentukan ID EC2 instance dari node yang terkait.

```
kubectl get pod <pod-name> -o wide
```

- Gunakan ID EC2 instance untuk mengambil output konsol.

```
aws ec2 get-console-output --instance-id <instance id> --latest --output text
```

Dapatkan log node dengan menggunakan wadah debug dan CLI **kubectl**

Cara yang disarankan untuk mengambil log dari node Mode Otomatis EKS adalah dengan menggunakan NodeDiagnostic sumber daya. Untuk langkah-langkah ini, lihat [the section called “Dapatkan log simpul”](#).

Namun, Anda dapat melakukan streaming log secara langsung dari sebuah instance dengan menggunakan `kubectl debug node` perintah. Perintah ini meluncurkan Pod baru pada node yang ingin Anda debug yang kemudian dapat Anda gunakan secara interaktif.

- Luncurkan wadah debug. Perintah berikut menggunakan `i-01234567890123456` ID instance node, `-it` mengalokasikan a tty dan melampirkan stdin untuk penggunaan interaktif, dan menggunakan `sysadmin` profil dari file `kubeconfig`.

```
kubectl debug node/i-01234567890123456 -it --profile=sysadmin --image=public.ecr.aws/amazonlinux/amazonlinux:2023
```

Contoh output adalah sebagai berikut.

```
Creating debugging pod node-debugger-i-01234567890123456-nxb9c with container
debugger on node i-01234567890123456.
If you don't see a command prompt, try pressing enter.
bash-5.2#
```

- Dari shell, Anda sekarang dapat menginstal `util-linux-core` yang menyediakan `nsenter` perintah. Gunakan `nsenter` untuk memasukkan namespace mount PID 1 (`init`) pada host, dan jalankan `journalctl` perintah untuk mengalirkan log dari: `kubelet`

```
yum install -y util-linux-core
nsenter -t 1 -m journalctl -f -u kubelet
```

Untuk keamanan, image container Amazon Linux tidak menginstal banyak binari secara default. Anda dapat menggunakan `yum whatprovides` perintah untuk mengidentifikasi paket yang harus diinstal untuk menyediakan biner yang diberikan.

```
yum whatprovides ps
```

```
Last metadata expiration check: 0:03:36 ago on Thu Jan 16 14:49:17 2025.
procps-ng-3.3.17-1.amzn2023.0.2.x86_64 : System and process monitoring utilities
Repo          : @System
Matched from:
Filename      : /usr/bin/ps
Provide       : /bin/ps

procps-ng-3.3.17-1.amzn2023.0.2.x86_64 : System and process monitoring utilities
Repo          : amazonlinux
Matched from:
Filename      : /usr/bin/ps
Provide       : /bin/ps
```

Melihat sumber daya yang terkait dengan Mode Otomatis EKS di AWS Konsol

Anda dapat menggunakan AWS konsol untuk melihat status sumber daya yang terkait dengan kluster Mode Otomatis EKS Anda.

- [Volume EBS](#)
 - Lihat volume Mode Otomatis EKS dengan mencari kunci tag `eks:eks-cluster-name`
- [Penyeimbang Beban](#)
 - Lihat penyeimbang beban Mode Otomatis EKS dengan mencari kunci tag `eks:eks-cluster-name`
- [EC2 Contoh](#)
 - Lihat contoh Mode Otomatis EKS dengan mencari kunci tag `eks:eks-cluster-name`

Lihat Kesalahan IAM di akun Anda AWS

1. Arahkan ke CloudTrail konsol
2. Pilih “Riwayat Acara” dari panel navigasi kiri

3. Terapkan filter kode kesalahan:

- AccessDenied
- UnauthorizedOperation
- InvalidClientTokenId

Cari kesalahan yang terkait dengan cluster EKS Anda. Gunakan pesan kesalahan untuk memperbarui entri akses EKS, peran IAM cluster, atau peran IAM node. Anda mungkin perlu melampirkan kebijakan baru ke peran ini dengan izin untuk Mode Otomatis EKS.

Memecahkan masalah Pod yang gagal menjadwalkan ke node Mode Otomatis

Jika pod tetap dalam Pending status dan tidak dijadwalkan ke node mode otomatis, verifikasi apakah pod atau manifes penerapan Anda memiliki nodeSelector file. Jika nodeSelector ada, pastikan bahwa itu digunakan `eks.amazonaws.com/compute-type: auto` untuk dijadwalkan pada node yang dibuat oleh EKS Auto Mode. Untuk informasi selengkapnya tentang label node yang digunakan oleh Mode Otomatis EKS, lihat [the section called “Kontrol penyebaran”](#).

Memecahkan masalah node yang tidak bergabung dengan cluster

Mode Otomatis EKS secara otomatis mengonfigurasi EC2 instance baru dengan informasi yang benar untuk bergabung dengan cluster, termasuk titik akhir cluster dan otoritas sertifikat klaster (CA). Namun, instance ini masih bisa gagal untuk bergabung dengan cluster EKS sebagai node. Jalankan perintah berikut untuk mengidentifikasi instance yang tidak bergabung dengan cluster:

1. `kubectl get nodeclaim`Jalankan untuk memeriksa NodeClaims ituReady = False.

```
kubectl get nodeclaim
```

2. Jalankan `kubectl describe nodeclaim <node_claim>` dan lihat di bawah Status untuk menemukan masalah apa pun yang mencegah node bergabung dengan cluster.

```
kubectl describe nodeclaim <node_claim>
```

Pesan kesalahan umum:

Error getting launch template configs

Anda mungkin menerima kesalahan ini jika Anda menyetel tag kustom NodeClass dengan izin peran IAM cluster default. Lihat [the section called “Identitas dan akses”](#).

Error creating fleet

Mungkin ada beberapa masalah otorisasi dengan memanggil RunInstances panggilan dari EC2 API. AWS CloudTrail Periksa kesalahan dan lihat [the section called “Mode Otomatis klaster peran IAM”](#) izin IAM yang diperlukan.

Mendeteksi masalah konektivitas node dengan **VPC Reachability Analyzer**

Note

Anda dikenakan biaya untuk setiap analisis yang menjalankan VPC Reachability Analyzer. Untuk detail harga, lihat Harga [Amazon VPC](#).

Salah satu alasan mengapa instance tidak bergabung dengan cluster adalah masalah konektivitas jaringan yang mencegah mereka mencapai server API. Untuk mendiagnosis masalah ini, Anda dapat menggunakan [VPC Reachability Analyzer](#) untuk melakukan analisis konektivitas antara node yang gagal bergabung dengan cluster dan server API. Anda akan membutuhkan dua informasi:

- ID instance dari node yang tidak dapat bergabung dengan cluster
- Alamat IP dari titik akhir server API Kubernetes

Untuk mendapatkan ID instance, Anda harus membuat beban kerja di cluster untuk menyebabkan Mode Otomatis EKS meluncurkan EC2 instance. Ini juga membuat NodeClaim objek di cluster Anda yang akan memiliki ID instance. Jalankan `kubectl get nodeclaim -o yaml` untuk mencetak semua yang NodeClaims ada di cluster Anda. Masing-masing NodeClaim berisi ID instance sebagai bidang dan lagi di providerId:

```
kubectl get nodeclaim -o yaml
```

Contoh output adalah sebagai berikut.

```
nodeName: i-01234567890123456
```

```
providerID: aws:///us-west-2a/i-01234567890123456
```

Anda dapat menentukan titik akhir server API Kubernetes dengan menjalankannya. `kubectl get endpoint kubernetes -o yaml` Alamat ada di bidang alamat:

```
kubectl get endpoints kubernetes -o yaml
```

Contoh output adalah sebagai berikut.

```
apiVersion: v1
kind: Endpoints
metadata:
  name: kubernetes
  namespace: default
subsets:
- addresses:
  - ip: 10.0.143.233
  - ip: 10.0.152.17
  ports:
  - name: https
    port: 443
    protocol: TCP
```

Dengan dua informasi ini, Anda dapat melakukan analisis s. Pertama-tama navigasikan ke VPC Reachability Analyzer di file.AWS Management Console

1. Klik “Buat dan Analisis Jalur”
2. Berikan nama untuk analisis (misalnya “Node Join Failure”)
3. Untuk “Jenis Sumber” pilih “Instans”
4. Masukkan ID instance dari Node yang gagal sebagai “Sumber”
5. Untuk “Path Destination” pilih “IP Address”
6. Masukkan salah satu alamat IP untuk server API sebagai “Alamat Tujuan”
7. Perluas “Bagian Konfigurasi Header Paket Tambahan”
8. Masukkan “Port Tujuan” dari 443
9. Pilih “Protocol” sebagai TCP jika belum dipilih
10. Klik “Buat dan Analisis Jalur”

11 Analisis mungkin memakan waktu beberapa menit untuk menyelesaikannya. Jika hasil analisis menunjukkan kemampuan jangkauan yang gagal, itu akan menunjukkan di mana kegagalan berada di jalur jaringan sehingga Anda dapat menyelesaikan masalah.

Berbagi Volume di Seluruh Pod

Node Mode Otomatis EKS dikonfigurasi dengan SELinux mode penegakan yang memberikan lebih banyak isolasi antara Pod yang berjalan pada Node yang sama. Ketika SELinux diaktifkan, sebagian besar pod yang tidak memiliki hak istimewa akan secara otomatis memiliki label keamanan multi-kategori (MCS) mereka sendiri yang diterapkan padanya. Label MCS ini unik per Pod, dan dirancang untuk memastikan bahwa proses dalam satu Pod tidak dapat memanipulasi proses di Pod lain atau pada host. Bahkan jika Pod berlabel berjalan sebagai root dan memiliki akses ke sistem file host, ia tidak akan dapat memanipulasi file, membuat panggilan sistem sensitif pada host, mengakses runtime container, atau mendapatkan materi kunci rahasia kubelet.

Karena itu, Anda mungkin mengalami masalah saat mencoba berbagi data antar Pod. Misalnya, `PersistentVolumeClaim` dengan mode akses masih tidak `ReadWriteOnce` akan mengizinkan beberapa Pod untuk mengakses volume secara bersamaan.

Untuk mengaktifkan berbagi antar Pod ini, Anda dapat menggunakan `Pod seLinuxOptions` untuk mengonfigurasi label MCS yang sama pada Pod tersebut. Dalam contoh ini, kita menetapkan tiga kategori `c123`, `c456`, `c789` ke Pod. Ini tidak akan bertentangan dengan kategori apa pun yang ditetapkan ke Pod pada node secara otomatis, karena mereka hanya akan diberikan dua kategori.

```
securityContext:
  seLinuxOptions:
    level: "s0:c123,c456,c789"
```

Memecahkan masalah pengontrol yang disertakan dalam Mode Otomatis

Jika Anda memiliki masalah dengan pengontrol, Anda harus meneliti:

- Jika sumber daya yang terkait dengan pengontrol itu diformat dengan benar dan valid.
- Jika sumber daya AWS IAM dan Kubernetes RBAC dikonfigurasi dengan benar untuk kluster Anda. Untuk informasi selengkapnya, lihat [the section called “Identitas dan akses”](#).

Tinjau catatan rilis Mode Otomatis EKS

Halaman ini mendokumentasikan pembaruan ke Amazon EKS Auto Mode. Anda dapat memeriksa halaman ini secara berkala untuk pengumuman tentang fitur, perbaikan bug, masalah yang diketahui, dan fungsionalitas usang.

Untuk menerima pemberitahuan semua perubahan file sumber ke halaman dokumentasi khusus ini, Anda dapat berlangganan URL berikut dengan pembaca RSS:

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/automode/
auto-change.adoc.atom
```

Juni 30, 2025

Fitur: Mode Otomatis NodeClass sekarang menggunakan kunci KMS kustom yang dikonfigurasi untuk mengenkripsi volume root read-only instance, selain volume data. read/write Sebelumnya, kunci KMS khusus hanya digunakan untuk mengenkripsi volume data.

Juni 20, 2025

Fitur: Dukungan untuk mengontrol penyebaran beban kerja ke Reservasi Kapasitas EC2 Sesuai Permintaan (). ODCRs Ini menambahkan kunci opsional `capacityReservationSelectorTerms` ke NodeClass, memungkinkan Anda untuk secara eksplisit mengontrol beban kerja ODCRs Anda yang digunakan. Untuk informasi selengkapnya, lihat [the section called “Kontrol Penerapan ODCR”](#).

Juni 13, 2025

Fitur: Support untuk subnet pod terpisah di file. NodeClass Ini menambahkan kunci opsional `podSubnetSelectorTerms` dan `podSecurityGroupSelectorTerms` untuk mengatur subnet dan grup keamanan untuk pod. Untuk informasi selengkapnya, lihat [the section called “Pemilihan subnet untuk Pod”](#).

April 30, 2025

Fitur: Support untuk proxy jaringan maju di. NodeClass Ini menambahkan kunci opsional `advancedNetworking` untuk mengatur proxy HTTPS Anda. Untuk informasi selengkapnya, lihat [the section called “Spesifikasi Kelas Node”](#).

April 18, 2025

Fitur: Dukungan untuk menyelesaikan domain.local (biasanya dicadangkan untuk Multicast DNS) melalui DNS unicast.

April 11, 2025

Fitur: Ditambahkan `certificateBundles` dan `ephemeralStorage.kmsKeyID` ke `NodeClass`. Untuk informasi selengkapnya, lihat [the section called “Spesifikasi Kelas Node”](#).

Fitur: Peningkatan kecepatan tarik gambar, terutama untuk tipe instance dengan penyimpanan instans lokal yang dapat memanfaatkan dekompresi gambar yang lebih cepat.

Perbaikan Bug: Menyelesaikan kondisi balapan yang menyebabkan `FailedCreatePodSandBox`, Kesalahan saat melakukan panggilan: `dial tcp 127.0.0.1:50051: connect: koneksi terkadang ditolak` untuk penjadwalan Pod ke Node segera saat startup.

April 4, 2025

Fitur: Tingkatkan `registryPullQPS` dari 5 menjadi 25 dan `registryBurst` dari 10 menjadi 50 untuk mengurangi pelambatan tarik gambar yang diterapkan klien `() Failed to pull image xyz: pull QPS exceeded`

Maret 31, 2025

Perbaikan Bug: Memperbaiki masalah di mana jika Core DNS Pod berjalan pada node Mode Otomatis, kueri DNS dari Pod pada node akan mengenai Core DNS Pod alih-alih server DNS lokal node. Kueri DNS dari Pod pada node Mode Otomatis akan selalu masuk ke DNS lokal node.

Maret 21, 2025

Perbaikan Bug: Node Mode Otomatis sekarang diselesaikan `kube-dns.kube-system.svc.cluster.local` dengan benar ketika tidak ada `kube-dns` layanan yang diinstal di cluster. Mengatasi GitHub masalah [#2546](#).

Maret 14, 2025

Fitur: IPv4 jalan keluar diaktifkan dalam IPv6 cluster. IPv4 lalu lintas yang keluar dari kluster Mode IPv6 Otomatis sekarang akan secara otomatis diterjemahkan ke v4 alamat ENI primer node.

Mengatur beban kerja dengan kluster Amazon EKS

Sebuah kluster Amazon EKS terdiri dari dua komponen utama:

- Bidang pengendali Amazon EKS
- Simpul Amazon EKS yang terdaftar dengan bidang kontrol

Bidang pengendali Amazon EKS terdiri dari simpul bidang pengendali yang menjalankan perangkat lunak Kubernetes, seperti etcd dan server Kubernetes API. Control plane berjalan di akun yang dikelola oleh AWS, dan Kubernetes API diekspos melalui endpoint Amazon EKS yang terkait dengan cluster Anda. Setiap bidang kontrol cluster Amazon EKS adalah penyewa tunggal dan unik, dan berjalan pada rangkaian instans EC2 Amazonnya sendiri.

Semua data yang disimpan oleh etcd node dan volume Amazon EBS terkait dienkripsi menggunakan KMS. AWS Control plane kluster disediakan di beberapa Availability Zone dan dihadapkan oleh Penyeimbang Beban Jaringan Elastic Load Balancing. Amazon EKS juga menyediakan antarmuka jaringan elastis di subnet VPC Anda untuk menyediakan konektivitas dari instance bidang kontrol ke node (misalnya, untuk `kubectl exec logs proxy` mendukung aliran data).

Important

Di lingkungan Amazon EKS, etcd penyimpanan dibatasi hingga 8 GiB sesuai panduan [hulu](#). Anda dapat memantau metrik untuk ukuran database saat ini dengan menjalankan perintah berikut. Jika kluster Anda memiliki versi Kubernetes di bawah `1.28`, ganti dengan `apiserver_storage_size_bytes` `apiserver_storage_db_total_size_in_bytes`

```
kubectl get --raw=/metrics | grep "apiserver_storage_size_bytes"
```

Node Amazon EKS berjalan di AWS akun Anda dan terhubung ke bidang kontrol kluster Anda melalui titik akhir server API dan file sertifikat yang dibuat untuk kluster Anda.

 Note

- Anda dapat mengetahui bagaimana berbagai komponen Amazon EKS bekerja di [Konfigurasi jaringan untuk kluster Amazon EKS](#).
- Untuk cluster yang terhubung, lihat [Konektor Amazon EKS](#).

Topik

- [Buat kluster Mode Otomatis Amazon EKS](#)
- [Buat kluster Amazon EKS](#)
- [Bersiaplah untuk upgrade versi Kubernetes dan pecahkan masalah kesalahan konfigurasi dengan wawasan kluster](#)
- [Perbarui klaster yang ada ke versi Kubernetes baru](#)
- [Hapus klaster](#)
- [Titik akhir server API cluster](#)
- [Menyebarkan node Windows pada kluster EKS](#)
- [Nonaktifkan dukungan Windows](#)
- [Menyebarkan kluster pribadi dengan akses internet terbatas](#)
- [Memahami siklus hidup versi Kubernetes di EKS](#)
- [Lihat versi platform Amazon EKS untuk setiap versi Kubernetes](#)
- [Komputasi klaster skala dengan Karpenter dan Cluster Autoscaler](#)
- [Pelajari tentang Pergeseran Zonal Pengontrol Pemulihan Aplikasi Amazon \(ARC\) di Amazon EKS](#)
- [Aktifkan EKS Zonal Shift untuk menghindari gangguan Availability Zone](#)

Buat kluster Mode Otomatis Amazon EKS

Topik ini memberikan petunjuk terperinci untuk membuat kluster Mode Otomatis Amazon EKS menggunakan opsi konfigurasi lanjutan. Ini mencakup prasyarat, opsi jaringan, dan konfigurasi add-on. Prosesnya meliputi pengaturan peran IAM, mengonfigurasi pengaturan cluster, menentukan parameter jaringan, dan memilih add-on. Pengguna dapat membuat cluster menggunakan CLI AWS Management Console atau, step-by-step dengan panduan yang disediakan untuk kedua metode.

AWS

Untuk pengguna yang mencari proses penyiapan yang kurang kompleks, lihat berikut ini untuk langkah-langkah pembuatan kluster yang disederhanakan:

- [the section called “CLI eksctl”](#)
- [the section called “ AWS CLI”](#)
- [the section called “Konsol manajemen”](#)

Panduan konfigurasi lanjutan ini ditujukan untuk pengguna yang memerlukan kontrol lebih terperinci atas pengaturan kluster Mode Otomatis EKS mereka dan terbiasa dengan konsep dan persyaratan Amazon EKS. Sebelum melanjutkan dengan konfigurasi lanjutan, pastikan Anda telah memenuhi semua prasyarat dan memiliki pemahaman menyeluruh tentang jaringan dan persyaratan IAM untuk kluster Mode Otomatis EKS.

Mode Otomatis EKS memerlukan izin IAM tambahan. Untuk informasi selengkapnya, lihat:

- [the section called “Peran IAM untuk Kluster Mode Otomatis EKS”](#)
- [the section called “Identitas dan akses”](#)

Note

Jika Anda ingin membuat cluster tanpa Mode Otomatis EKS, lihat [the section called “Membuat kluster”](#).

Topik ini mencakup konfigurasi lanjutan. Jika Anda ingin memulai dengan Mode Otomatis EKS, lihat [the section called “Buat cluster”](#).

Prasyarat

- VPC dan subnet yang ada yang memenuhi persyaratan [Amazon EKS](#). Sebelum Anda menerapkan kluster untuk penggunaan produksi, kami sarankan Anda memiliki pemahaman menyeluruh tentang persyaratan VPC dan subnet. Jika Anda tidak memiliki VPC dan subnet, Anda dapat membuatnya menggunakan template yang disediakan [Amazon EKS](#). AWS CloudFormation
- Alat baris `kubectl` perintah diinstal pada perangkat Anda atau AWS CloudShell. Versinya bisa sama dengan atau hingga satu versi minor lebih awal atau lebih lambat dari versi Kubernetes dari kluster Anda. Misalnya, jika versi cluster Anda `1.29`, Anda dapat menggunakan `kubectl`

versi 1.28, 1.29, atau 1.30 dengan itu. Untuk menginstal atau memutakhirkan kubectl, lihat [the section called “Siapkan kubectl dan eksctl”](#).

- Versi 2.12.3 atau yang lebih baru atau versi 1.27.160 atau yang lebih baru dari AWS Command Line Interface (AWS CLI) diinstal dan dikonfigurasi pada perangkat Anda atau AWS CloudShell. Untuk memeriksa versi Anda saat ini, gunakan `aws --version`. Untuk menginstal versi terbaru, lihat [Menginstal](#) dan [Konfigurasi cepat dengan aws configure](#) di Panduan Pengguna Antarmuka Baris AWS Perintah.
- [Prinsipal IAM](#) dengan izin untuk membuat dan memodifikasi sumber daya EKS dan IAM.

Buat cluster - AWS konsol

1. Buka [konsol Amazon EKS](#).
2. Pilih Add cluster dan kemudian pilih Create.
3. Di bawah Opsi konfigurasi, pilih Konfigurasi kustom.
 - Topik ini mencakup konfigurasi kustom. Untuk informasi tentang Konfigurasi cepat, lihat [the section called “Konsol manajemen”](#).
4. Konfirmasikan Gunakan Mode Otomatis EKS diaktifkan.
 - Topik ini mencakup pembuatan cluster dengan Mode Otomatis EKS. Untuk informasi selengkapnya tentang membuat cluster tanpa Mode Otomatis EKS, lihat [the section called “Membuat klaster”](#).
5. Pada halaman Configure cluster, masukkan bidang-bidang berikut:
 - Nama — Nama untuk klaster Anda. Nama hanya dapat berisi karakter alfanumerik (peka huruf besar/kecil), tanda hubung, dan garis bawah. Itu harus dimulai dengan karakter alfanumerik dan tidak boleh lebih dari 100 karakter. Nama harus unik di dalam AWS Wilayah dan AWS akun tempat Anda membuat klaster.
 - Peran IAM Cluster — Pilih peran IAM klaster Amazon EKS yang Anda buat untuk memungkinkan bidang kontrol Kubernetes mengelola AWS sumber daya atas nama Anda. Jika sebelumnya Anda belum membuat peran IAM Cluster untuk Mode Otomatis EKS, pilih tombol Buat peran yang disarankan untuk membuat peran dengan izin yang diperlukan di konsol IAM.
 - Versi Kubernetes — Versi Kubernetes untuk digunakan untuk klaster Anda. Sebaiknya pilih versi terbaru, kecuali jika Anda memerlukan versi sebelumnya.
 - Kebijakan upgrade — Kebijakan versi Kubernetes yang ingin Anda tetapkan untuk klaster Anda. Jika Anda ingin klaster Anda hanya berjalan pada versi dukungan standar, Anda dapat memilih Standar. Jika Anda ingin klaster Anda memasukkan dukungan yang diperluas di akhir

dukungan standar untuk sebuah versi, Anda dapat memilih Extended. Jika Anda memilih versi Kubernetes yang saat ini dalam dukungan diperpanjang, Anda tidak dapat memilih dukungan standar sebagai opsi.

6. Di bagian Komputasi Mode Otomatis pada halaman cluster konfigurasi, masukkan bidang berikut:

- Node pool — Tentukan apakah Anda ingin menggunakan build in node pool. Untuk informasi selengkapnya, lihat [the section called “Tinjau kumpulan node bawaan”](#).
- Peran IAM Node - Jika Anda mengaktifkan salah satu kumpulan node bawaan, Anda harus memilih Peran IAM Node. Mode Otomatis EKS akan menetapkan peran ini ke node baru. Anda tidak dapat mengubah nilai ini setelah cluster dibuat. Jika sebelumnya Anda belum membuat peran IAM Node untuk Mode Otomatis EKS, pilih tombol Buat peran yang disarankan untuk membuat peran dengan izin yang diperlukan. Untuk informasi selengkapnya tentang peran ini, silakan lihat [the section called “Identitas dan akses”](#).

7. Di bagian akses Cluster pada halaman cluster konfigurasi, masukkan bidang berikut:

- Akses administrator klaster Bootstrap - Pembuat klaster secara otomatis adalah administrator Kubernetes. Jika Anda ingin menonaktifkan ini, pilih Larang akses administrator klaster.
- Mode otentikasi cluster — Mode Otomatis EKS memerlukan entri akses EKS, mode otentikasi EKS API. Anda dapat mengaktifkan mode ConfigMap otentikasi secara opsional dengan memilih EKS API dan. ConfigMap

8. Masukkan bidang yang tersisa di halaman konfigurasi cluster:

- Enkripsi rahasia — (Opsional) Pilih untuk mengaktifkan enkripsi rahasia Kubernetes menggunakan kunci KMS. Anda juga dapat mengaktifkan ini setelah Anda membuat cluster Anda. Sebelum Anda mengaktifkan kemampuan ini, pastikan Anda sudah terbiasa dengan informasi di dalamnya [the section called “Aktifkan enkripsi rahasia”](#).
- ARC Zonal shift — Mode Otomatis EKS tidak mendukung pergeseran Arc Zonal.
- Tanda — (Opsional) Tambahkan tanda apapun ke klaster Anda. Untuk informasi selengkapnya, lihat [the section called “Pemberian tag pada sumber daya Anda”](#).

Setelah selesai dengan halaman ini, pilih Berikutnya.

9. Pada halaman Tentukan jaringan, pilih nilai untuk kolom berikut:

- VPC — Pilih VPC yang sudah ada yang memenuhi [persyaratan Amazon EKS VPC](#) untuk membuat klaster Anda. Sebelum memilih VPC, kami sarankan Anda memahami semua persyaratan dan pertimbangan di dalamnya. [the section called “Persyaratan VPC dan subnet”](#) Anda tidak dapat mengubah VPC mana yang ingin Anda gunakan setelah pembuatan cluster.

Jika tidak VPCs terdaftar, maka Anda harus membuatnya terlebih dahulu. Untuk informasi selengkapnya, lihat [the section called “Buat VPC”](#).

- Subnet — Secara default, semua subnet yang tersedia di VPC yang ditentukan di bidang sebelumnya telah dipilih sebelumnya. Anda harus memilih setidaknya dua.

Subnet yang Anda pilih harus memenuhi persyaratan [subnet Amazon EKS](#). Sebelum memilih subnet, sebaiknya Anda terbiasa dengan semua persyaratan dan pertimbangan [VPC Amazon EKS dan subnet](#).

Grup keamanan — (Opsional) Tentukan satu atau beberapa grup keamanan yang ingin Anda kaitkan Amazon EKS ke antarmuka jaringan yang dibuatnya.

Apakah Anda memilih grup keamanan atau tidak, Amazon EKS membuat grup keamanan yang memungkinkan komunikasi antara cluster dan VPC Anda. Amazon EKS mengaitkan grup keamanan ini, dan apa pun yang Anda pilih, ke antarmuka jaringan yang dibuatnya. Untuk informasi selengkapnya tentang grup keamanan klaster yang dibuat Amazon EKS, lihat [the section called “Persyaratan grup keamanan”](#). Anda dapat mengubah aturan di grup keamanan klaster yang dibuat Amazon EKS.

- Pilih keluarga alamat IP cluster — Anda dapat memilih salah satu IPv4 dan IPv6.

Kubernetes memberikan IPv4 alamat ke Pod dan layanan, secara default. Sebelum memutuskan untuk menggunakan IPv6 keluarga, pastikan Anda sudah terbiasa dengan semua pertimbangan dan persyaratan dalam persyaratan dan pertimbangan [VPCthe section called “Persyaratan dan pertimbangan subnet”, the section called “Persyaratan grup keamanan”, dan topik. the section called “IPv6”](#) Jika Anda memilih IPv6 keluarga, Anda tidak dapat menentukan rentang alamat untuk Kubernetes untuk menetapkan alamat IPv6 layanan seperti yang Anda bisa untuk keluarga. IPv4 Kubernetes memberikan alamat layanan dari rentang alamat lokal yang unik (). `fc00::/7`

- (Opsional) Pilih Konfigurasi rentang alamat IP Layanan Kubernetes dan tentukan rentang Layanan. **IPv4**

Menentukan jangkauan Anda sendiri dapat membantu mencegah konflik antara layanan Kubernetes dan jaringan lain yang di-peer atau terhubung ke VPC Anda. Masukkan rentang dalam notasi CIDR. Sebagai contoh: `10.2.0.0/16`.

Blok CIDR harus memenuhi persyaratan berikut:

- Berada dalam salah satu rentang berikut: `10.0.0.0/8`, `172.16.0.0/12`, atau `192.168.0.0/16`.

- Memiliki ukuran minimum /24 dan ukuran maksimum/12.
- Tidak tumpang tindih dengan kisaran VPC untuk sumber daya Amazon EKS Anda.

Anda hanya dapat menentukan opsi ini saat menggunakan keluarga IPv4 alamat dan hanya pada pembuatan cluster. Jika Anda tidak menentukan ini, maka Kubernetes memberikan alamat IP layanan dari blok atau CIDR. 10.100.0.0/16 172.20.0.0/16

- Untuk akses endpoint Cluster, pilih opsi. Setelah cluster Anda dibuat, Anda dapat mengubah opsi ini. Sebelum memilih opsi non-default, pastikan untuk membiasakan diri dengan opsi dan implikasinya. Untuk informasi selengkapnya, lihat [the section called “Akses titik akhir kluster”](#).

Setelah selesai dengan halaman ini, pilih Berikutnya.

10(Opsional) Pada halaman Konfigurasi observabilitas, pilih opsi pencatatan bidang Metrik dan Kontrol mana yang akan diaktifkan. Secara default, setiap jenis log dimatikan.

- Untuk informasi selengkapnya tentang opsi metrik Prometheus, lihat [the section called “Langkah 1: Aktifkan metrik Prometheus”](#)
- Untuk informasi selengkapnya tentang opsi Pencatatan bidang kontrol, lihat [the section called “Kontrol log pesawat”](#).
- Setelah selesai dengan halaman ini, pilih Berikutnya.

11 Pada halaman Pilih add-on, pilih add-on yang ingin Anda tambahkan ke cluster Anda. Anda dapat memilih add-on Amazon EKS dan add-on AWS Marketplace sebanyak yang Anda butuhkan. Jika add-on AWS Marketplace yang ingin Anda instal tidak terdaftar, Anda dapat mengklik penomoran halaman untuk melihat hasil halaman tambahan atau mencari add-on AWS Marketplace yang tersedia dengan memasukkan teks di kotak pencarian. Anda juga dapat memfilter berdasarkan kategori, vendor, atau model harga dan kemudian memilih add-on dari hasil pencarian. Saat membuat cluster, Anda dapat melihat, memilih, dan menginstal add-on apa pun yang mendukung EKS Pod Identities seperti yang dijelaskan di dalamnya. [the section called “Identitas Pod”](#)

- Mode Otomatis EKS mengotomatiskan fungsionalitas add-on tertentu. Jika Anda berencana untuk menerapkan Grup Node Terkelola EKS ke Kluster Mode Otomatis EKS Anda, pilih Add-on Amazon EKS Tambahan dan tinjau opsinya. Anda mungkin perlu menginstal add-on seperti CoreDNS dan kube-proxy. EKS hanya akan menginstal add-on di bagian ini pada node dan grup node yang dikelola sendiri.
- Setelah selesai dengan halaman ini, pilih Berikutnya.

12 Pada halaman Konfigurasi pengaturan add-on yang dipilih, pilih versi yang ingin Anda instal. Anda selalu dapat memperbarui ke versi yang lebih baru setelah pembuatan cluster.

Untuk add-on yang mendukung Identitas Pod EKS, Anda dapat menggunakan konsol untuk membuat peran secara otomatis dengan nama, kebijakan AWS terkelola, dan kebijakan kepercayaan yang telah diisi sebelumnya secara khusus untuk add-on. Anda dapat menggunakan kembali peran yang ada atau membuat peran baru untuk add-on yang didukung. Untuk langkah-langkah menggunakan konsol untuk membuat peran untuk add-on yang mendukung EKS Pod Identities, lihat [the section called “Buat add-on \(AWS Konsol\)”](#). Jika add-on tidak mendukung EKS Pod Identity, pesan akan ditampilkan dengan instruksi untuk menggunakan wizard untuk membuat peran IAM untuk akun layanan (IRSA) setelah cluster dibuat.

Anda dapat memperbarui konfigurasi setiap add-on setelah pembuatan cluster. Untuk informasi selengkapnya tentang mengonfigurasi add-on, lihat [the section called “Perbarui add-on”](#). Setelah selesai dengan halaman ini, pilih Berikutnya.

13 Pada halaman Tinjau dan buat, tinjau informasi yang Anda masukkan atau pilih pada halaman sebelumnya. Jika Anda perlu melakukan perubahan, pilih Edit. Saat Anda puas, pilih Buat. Bidang Status menunjukkan CREATING saat cluster disediakan.

Note

Anda mungkin menerima kesalahan bahwa salah satu Availability Zone dalam permintaan Anda tidak memiliki kapasitas yang cukup untuk membuat klaster Amazon EKS. Jika hal ini terjadi, output galat berisi Availability Zones yang dapat mendukung klaster baru. Cobalah untuk kembali membuat klaster dengan setidaknya dua subnet yang terletak di Availability Zones yang didukung untuk akun Anda. Untuk informasi selengkapnya, lihat [the section called “Kapasitas tidak mencukupi”](#).

Penyediaan klaster memerlukan waktu beberapa menit.

Buat cluster - AWS CLI

Instruksi CLI berikut mencakup pembuatan sumber daya IAM dan membuat cluster.

Membuat Peran IAM Kluster Mode Otomatis EKS

Langkah 1: Buat Kebijakan Kepercayaan

Buat kebijakan kepercayaan yang memungkinkan layanan Amazon EKS untuk mengambil peran. Simpan kebijakan sebagai `trust-policy.json`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

Langkah 2: Buat Peran IAM

Gunakan kebijakan kepercayaan untuk membuat Peran IAM Cluster:

```
aws iam create-role \
  --role-name AmazonEKSAutoClusterRole \
  --assume-role-policy-document file://trust-policy.json
```

Langkah 3: Perhatikan Peran ARN

Ambil dan simpan ARN dari peran baru untuk digunakan dalam langkah-langkah selanjutnya:

```
aws iam get-role --role-name AmazonEKSAutoClusterRole --query "Role.Arn" --output text
```

Langkah 4: Lampirkan Kebijakan yang Diperlukan

Lampirkan kebijakan AWS terkelola berikut ke Peran IAM Cluster untuk memberikan izin yang diperlukan:

EKSClusterKebijakan Amazon:

```
aws iam attach-role-policy \  
  --role-name AmazonEKSAutoClusterRole \  
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSClusterPolicy
```

EKSComputeKebijakan Amazon:

```
aws iam attach-role-policy \  
  --role-name AmazonEKSAutoClusterRole \  
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSComputePolicy
```

Amazon EKSBLOCK StoragePolicy:

```
aws iam attach-role-policy \  
  --role-name AmazonEKSAutoClusterRole \  
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSBLOCKStoragePolicy
```

Amazon EKSLoad BalancingPolicy:

```
aws iam attach-role-policy \  
  --role-name AmazonEKSAutoClusterRole \  
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSLoadBalancingPolicy
```

EKSNetworkingKebijakan Amazon:

```
aws iam attach-role-policy \  
  --role-name AmazonEKSAutoClusterRole \  
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSNetworkingPolicy
```

Buat Peran IAM Node Mode Otomatis EKS

Langkah 1: Buat Kebijakan Kepercayaan

Buat kebijakan kepercayaan yang memungkinkan layanan Amazon EKS untuk mengambil peran. Simpan kebijakan sebagai `node-trust-policy.json`:

```
{  
  "Version": "2012-10-17",  
  "Statement": [  

```



```
{
  "Effect": "Allow",
  "Principal": {
    "Service": "ec2.amazonaws.com"
  },
  "Action": "sts:AssumeRole"
}
```

Langkah 2: Buat Peran IAM Node

Gunakan `node-trust-policyfile.json` dari langkah sebelumnya untuk menentukan entitas mana yang dapat mengambil peran. Jalankan perintah berikut untuk membuat Peran IAM Node:

```
aws iam create-role \
  --role-name AmazonEKSAutoNodeRole \
  --assume-role-policy-document file:///node-trust-policy.json
```

Langkah 3: Perhatikan Peran ARN

Setelah membuat peran, ambil dan simpan ARN dari Peran IAM Node. Anda akan membutuhkan ARN ini pada langkah selanjutnya. Gunakan perintah berikut untuk mendapatkan ARN:

```
aws iam get-role --role-name AmazonEKSAutoNodeRole --query "Role.Arn" --output text
```

Langkah 4: Lampirkan Kebijakan yang Diperlukan

Lampirkan kebijakan AWS terkelola berikut ke Peran IAM Node untuk memberikan izin yang diperlukan:

Amazon EKSWorker NodeMinimalPolicy:

```
aws iam attach-role-policy \
  --role-name AmazonEKSAutoNodeRole \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodeMinimalPolicy
```

Amazon EC2 ContainerRegistryPullOnly:

```
aws iam attach-role-policy \
  --role-name AmazonEKSAutoNodeRole \
```

```
--policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryPullOnly
```

Buat cluster

1. Buat cluster Anda dengan perintah berikut. Sebelum menjalankan perintah, buat penggantian berikut:

- Ganti *region-code* dengan AWS Region tempat Anda ingin membuat cluster Anda.
- Ganti *my-cluster* dengan nama untuk cluster Anda. Nama hanya dapat berisi karakter alfanumerik (peka huruf besar/kecil), tanda hubung, dan garis bawah. Itu harus dimulai dengan karakter alfanumerik dan tidak boleh lebih dari 100 karakter. Nama harus unik di dalam AWS Wilayah dan AWS akun tempat Anda membuat klaster.
- Ganti *1.30* dengan [versi yang didukung Amazon EKS](#).
- Ganti *111122223333* dengan ID akun Anda
- Jika Anda telah membuat Peran IAM dengan nama berbeda untuk peran Cluster dan Node, ganti. ARNs
- Ganti nilainya `subnetIds` dengan nilai Anda sendiri. Anda juga dapat menambahkan tambahan IDs. Anda harus menentukan setidaknya dua subnet IDs.

Subnet yang Anda pilih harus memenuhi persyaratan [subnet Amazon EKS](#). Sebelum memilih subnet, sebaiknya Anda terbiasa dengan semua persyaratan dan pertimbangan [VPC Amazon EKS dan subnet](#).

- Jika Anda tidak ingin menentukan ID grup keamanan, hapus `, securityGroupIds=sg-
<ExampleID1>` dari perintah. Jika Anda ingin menentukan satu atau beberapa grup keamanan IDs, ganti nilainya `securityGroupIds` dengan milik Anda sendiri. Anda juga dapat menambahkan tambahan IDs.

Apakah Anda memilih grup keamanan atau tidak, Amazon EKS membuat grup keamanan yang memungkinkan komunikasi antara cluster dan VPC Anda. Amazon EKS mengaitkan grup keamanan ini, dan apa pun yang Anda pilih, ke antarmuka jaringan yang dibuatnya. Untuk informasi selengkapnya tentang grup keamanan klaster yang dibuat Amazon EKS, lihat [the section called “Persyaratan grup keamanan”](#). Anda dapat mengubah aturan di grup keamanan klaster yang dibuat Amazon EKS.

```
aws eks create-cluster \  
  --region region-code \  
  --name my-cluster \  
  --kubernetes-version 1.30 \  
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryPullOnly
```

```
--role-arn arn:aws:iam::111122223333:role/AmazonEKSAutoClusterRole \
--resources-vpc-config '{"subnetIds": ["subnet-ExampleID1", "subnet-ExampleID2"],
"securityGroupIds": ["sg-ExampleID1"], "endpointPublicAccess": true,
"endpointPrivateAccess": true}' \
--compute-config '{"enabled": true, "nodeRoleArn": "arn:aws:
iam::111122223333:role/AmazonEKSAutoNodeRole", "nodePools": ["general-purpose",
"system"]}' \
--kubernetes-network-config '{"elasticLoadBalancing": {"enabled": true}}' \
--storage-config '{"blockStorage": {"enabled": true}}' \
--access-config '{"authenticationMode": "API"}'
```

Note

Anda mungkin menerima kesalahan bahwa salah satu Availability Zone dalam permintaan Anda tidak memiliki kapasitas yang cukup untuk membuat kluster Amazon EKS. Jika hal ini terjadi, output galat berisi Availability Zones yang dapat mendukung kluster baru. Cobalah untuk kembali membuat kluster dengan setidaknya dua subnet yang terletak di Availability Zones yang didukung untuk akun Anda. Untuk informasi selengkapnya, lihat [the section called “Kapasitas tidak mencukupi”](#).

Berikut ini adalah pengaturan opsional yang, jika diperlukan, harus ditambahkan ke perintah sebelumnya. Anda hanya dapat mengaktifkan opsi ini saat Anda membuat cluster, bukan setelahnya.

- Jika Anda ingin menentukan dari mana IPv4 Classless Inter-domain Routing (CIDR) blok Kubernetes memberikan alamat IP layanan, Anda harus menentukannya dengan menambahkan ke perintah berikut. `--kubernetes-network-config serviceIpv4Cidr=<cidr-block>`

Menentukan jangkauan Anda sendiri dapat membantu mencegah konflik antara layanan Kubernetes dan jaringan lain yang di-peer atau terhubung ke VPC Anda. Masukkan rentang dalam notasi CIDR. Sebagai contoh: `10.2.0.0/16`.

Blok CIDR harus memenuhi persyaratan berikut:

- Berada dalam salah satu rentang berikut: `10.0.0.0/8`, `172.16.0.0/12`, atau `192.168.0.0/16`.
- Memiliki ukuran minimum `/24` dan ukuran maksimum `/12`.
- Tidak tumpang tindih dengan kisaran VPC untuk sumber daya Amazon EKS Anda.

Anda hanya dapat menentukan opsi ini saat menggunakan keluarga IPv4 alamat dan hanya pada pembuatan cluster. Jika Anda tidak menentukan ini, maka Kubernetes memberikan alamat IP layanan dari blok atau CIDR. 10.100.0.0/16 172.20.0.0/16

- Jika Anda membuat klaster dan ingin klaster menetapkan IPv6 alamat ke Pod dan layanan, bukan IPv4 alamat, tambahkan `--kubernetes-network-config ipFamily=ipv6` ke perintah berikut.

Kubernetes memberikan IPv4 alamat ke Pod dan layanan, secara default. Sebelum memutuskan untuk menggunakan IPv6 keluarga, pastikan Anda sudah terbiasa dengan semua pertimbangan dan persyaratan dalam persyaratan dan pertimbangan [VPCthe section called “Persyaratan dan pertimbangan subnet”](#), [the section called “Persyaratan grup keamanan”](#), dan topik. [the section called “IPv6”](#) Jika Anda memilih IPv6 keluarga, Anda tidak dapat menentukan rentang alamat untuk Kubernetes untuk menetapkan alamat IPv6 layanan seperti yang Anda bisa untuk keluarga. IPv4 Kubernetes memberikan alamat layanan dari rentang alamat lokal yang unik (). fc00::/7

2. Dibutuhkan beberapa menit untuk menyediakan cluster. Anda dapat melakukan kueri status klaster Anda dengan perintah berikut.

```
aws eks describe-cluster --region region-code --name my-cluster --query
"cluster.status"
```

Langkah selanjutnya

- [the section called “Akses klaster dengan kubectl”](#)
- [the section called “Berikan izin”](#)
- [Aktifkan enkripsi rahasia untuk cluster Anda.](#)
- [Konfigurasi logging untuk klaster Anda.](#)
- [Tambahkan node ke cluster Anda.](#)

Buat kluster Amazon EKS

Note

Topik ini mencakup pembuatan kluster EKS tanpa Mode Otomatis EKS.

Untuk petunjuk rinci tentang membuat kluster Mode Otomatis EKS, lihat [the section called “Buat kluster otomatis”](#).

Untuk memulai dengan Mode Otomatis EKS, lihat [the section called “Buat cluster \(Mode Otomatis EKS\)”](#).

Topik ini memberikan ikhtisar opsi yang tersedia dan menjelaskan apa yang harus dipertimbangkan saat Anda membuat kluster Amazon EKS. Jika Anda perlu membuat kluster dengan infrastruktur lokal sebagai komputasi untuk node, lihat [the section called “Buat cluster”](#). Jika ini adalah pertama kalinya Anda membuat cluster Amazon EKS, kami sarankan Anda mengikuti salah satu panduan kami di [Memulai](#). Panduan ini membantu Anda membuat cluster default yang sederhana tanpa memperluas ke semua opsi yang tersedia.

Prasyarat

- VPC dan subnet yang ada yang memenuhi persyaratan [Amazon](#) EKS. Sebelum Anda menerapkan kluster untuk penggunaan produksi, kami sarankan Anda memiliki pemahaman menyeluruh tentang persyaratan VPC dan subnet. Jika Anda tidak memiliki VPC dan subnet, Anda dapat membuatnya menggunakan template yang disediakan [Amazon EKS](#). AWS CloudFormation
- Alat baris `kubectl` perintah diinstal pada perangkat Anda atau AWS CloudShell. Versinya bisa sama dengan atau hingga satu versi minor lebih awal atau lebih lambat dari versi Kubernetes dari kluster Anda. Untuk menginstal atau memutakhirkan `kubectl`, lihat [the section called “Siapkan kubectl dan eksctl”](#).
- Versi 2.12.3 atau yang lebih baru atau versi 1.27.160 atau yang lebih baru dari AWS Command Line Interface (AWS CLI) diinstal dan dikonfigurasi pada perangkat Anda atau AWS CloudShell. Untuk memeriksa versi Anda saat ini, gunakan `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Package manager seperti `yum` atau `apt-get`, atau Homebrew untuk macOS seringkali merupakan beberapa versi di belakang versi terbaru CLI. AWS Untuk menginstal versi terbaru, lihat [Menginstal](#) dan [Konfigurasi cepat dengan aws configure](#) di Panduan Pengguna Antarmuka Baris AWS Perintah. Versi AWS CLI yang diinstal AWS CloudShell mungkin juga beberapa versi di belakang versi terbaru. Untuk memperbaruinya, lihat [Menginstal AWS CLI ke direktori home Anda](#) di AWS CloudShell Panduan Pengguna.
- [Prinsipal IAM](#) dengan izin `create` dan `kluster describe` Amazon EKS. Untuk informasi selengkapnya, lihat [the section called “Buat kluster Kubernetes lokal di Outpost”](#) dan [the section called “Buat daftar atau deskripsikan semua kluster”](#).

Langkah 1: Buat peran IAM cluster

1. Jika Anda sudah memiliki peran IAM cluster, atau Anda akan membuat cluster Anda dengan `eksctl`, maka Anda dapat melewati langkah ini. Secara default, `eksctl` buat peran untuk Anda.
2. Jalankan perintah berikut untuk membuat file JSON kebijakan kepercayaan IAM.

```
cat >eks-cluster-role-trust-policy.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

3. Buat peran IAM cluster Amazon EKS. Jika perlu, kata pengantar *eks-cluster-role-trust-policy.json* dengan jalur di komputer Anda tempat Anda menulis file pada langkah sebelumnya. Perintah tersebut mengaitkan kebijakan kepercayaan yang Anda buat pada langkah sebelumnya dengan peran tersebut. Untuk membuat peran IAM, [prinsipal IAM](#) yang membuat peran harus diberi `iam:CreateRole` tindakan (izin).

```
aws iam create-role --role-name myAmazonEKSClusterRole --assume-role-policy-document
file://"eks-cluster-role-trust-policy.json"
```

4. Anda dapat menetapkan kebijakan terkelola Amazon EKS atau membuat kebijakan kustom Anda sendiri. Untuk izin minimum yang harus Anda gunakan dalam kebijakan kustom, lihat [the section called "IAM role klaster"](#).

Lampirkan kebijakan terkelola Amazon EKS bernama [EKSClusterKebijakan Amazon](#) ke peran tersebut. Untuk melampirkan kebijakan IAM ke kepala sekolah [IAM, prinsipal](#) yang melampirkan kebijakan harus diberikan salah satu tindakan IAM berikut (izin): atau. `iam:AttachUserPolicy` `iam:AttachRolePolicy`

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSClusterPolicy --role-name myAmazonEKSClusterRole
```

Peran Tertaut Layanan

Amazon EKS secara otomatis membuat peran terkait layanan yang disebut `AWSServiceRoleForAmazonEKS`.

Ini adalah tambahan untuk peran IAM cluster. Peran tertaut layanan adalah jenis IAM role unik yang terkait langsung dengan Amazon EKS. Peran ini memungkinkan Amazon EKS untuk mengelola cluster di akun Anda. Untuk informasi selengkapnya, lihat [the section called “Peran cluster”](#).

Identitas IAM yang Anda gunakan untuk membuat kluster EKS harus memiliki izin untuk membuat peran terkait layanan. Ini termasuk `iam:CreateServiceLinkedRole` izin.

Jika peran terkait layanan belum ada, dan peran IAM Anda saat ini tidak memiliki izin yang cukup untuk membuatnya, operasi pembuatan kluster akan gagal.

Langkah 2: Buat cluster

Anda dapat membuat cluster dengan menggunakan:

- [eksctl](#)
- [yang AWS Management Console](#)
- [AWS CLI](#)

Buat cluster - eksctl

1. Anda memerlukan versi `0.210.0` atau yang lebih baru dari alat baris perintah `eksctl` yang diinstal pada perangkat Anda atau AWS CloudShell. Untuk menginstal atau memperbarui `eksctl`, lihat [Instalasi](#) dalam `eksctl` dokumentasi.
2. Buat IPv4 kluster Amazon EKS dengan versi Kubernetes default Amazon EKS di Region default Anda. AWS Sebelum menjalankan perintah, buat penggantian berikut:
3. Ganti `region-code` dengan AWS Region tempat Anda ingin membuat cluster Anda.
4. Ganti `my-cluster` dengan nama untuk cluster Anda. Nama hanya dapat berisi karakter alfanumerik (peka huruf besar/kecil) dan tanda hubung. Itu harus dimulai dengan karakter

alfanumerik dan tidak boleh lebih dari 100 karakter. Nama harus unik di dalam AWS Wilayah dan AWS akun tempat Anda membuat klaster.

5. Ganti **1.33** dengan [versi yang didukung Amazon EKS](#).
6. Ubah nilai `vpc-private-subnets` untuk memenuhi kebutuhan Anda. Anda juga dapat menambahkan tambahan IDs. Anda harus menentukan setidaknya dua subnet IDs. Jika Anda lebih suka menentukan subnet publik, Anda dapat mengubah `--vpc-private-subnets` ke `--vpc-public-subnets`. Subnet publik memiliki tabel rute terkait dengan rute ke gateway internet, tetapi subnet pribadi tidak memiliki tabel rute terkait. Sebaiknya gunakan subnet pribadi bila memungkinkan.

Subnet yang Anda pilih harus memenuhi persyaratan [subnet Amazon EKS](#). Sebelum memilih subnet, sebaiknya Anda terbiasa dengan semua persyaratan dan pertimbangan [VPC Amazon EKS dan subnet](#).

7. Jalankan perintah berikut:

```
eksctl create cluster --name my-cluster --region region-code --version 1.33 --vpc-private-subnets subnet-ExampleID1,subnet-ExampleID2 --without-nodegroup
```

Penyediaan klaster memerlukan waktu beberapa menit. Saat cluster sedang dibuat, beberapa baris output muncul. Baris terakhir output mirip dengan baris contoh berikut.

```
[#] EKS cluster "my-cluster" in "region-code" region is ready
```

8. Lanjutkan dengan [the section called “Langkah 3: Perbarui kubeconfig”](#)

Pengaturan Opsional

Untuk melihat sebagian besar opsi yang dapat Anda tentukan saat membuat cluster `eksctl`, gunakan `eksctl create cluster --help` perintah. Untuk melihat semua opsi yang tersedia, Anda dapat menggunakan config file. Untuk informasi selengkapnya, lihat [Menggunakan file config](#) dan [skema file config](#) di dokumentasi `eksctl`. Anda dapat menemukan [contoh file konfigurasi](#) di GitHub.

Berikut ini adalah pengaturan opsional yang, jika diperlukan, harus ditambahkan ke perintah sebelumnya. Anda hanya dapat mengaktifkan opsi ini saat membuat cluster, bukan setelahnya. Jika Anda perlu menentukan opsi ini, Anda harus membuat cluster dengan [file konfigurasi eksctl](#) dan menentukan pengaturan, daripada menggunakan perintah sebelumnya.

- Jika Anda ingin menentukan satu atau beberapa grup keamanan yang ditetapkan Amazon EKS ke antarmuka jaringan yang dibuatnya, tentukan opsi [SecurityGroup](#).

Apakah Anda memilih grup keamanan atau tidak, Amazon EKS membuat grup keamanan yang memungkinkan komunikasi antara cluster dan VPC Anda. Amazon EKS mengaitkan grup keamanan ini, dan apa pun yang Anda pilih, ke antarmuka jaringan yang dibuatnya. Untuk informasi selengkapnya tentang grup keamanan klaster yang dibuat Amazon EKS, lihat [the section called “Persyaratan grup keamanan”](#). Anda dapat mengubah aturan di grup keamanan klaster yang dibuat Amazon EKS.

- [Jika Anda ingin menentukan blok IPv4 Classless Inter-domain Routing \(CIDR\) mana Kubernetes memberikan alamat IP layanan, tentukan opsi CIDR layanan. IPv4](#)

Menentukan jangkauan Anda sendiri dapat membantu mencegah konflik antara layanan Kubernetes dan jaringan lain yang di-peer atau terhubung ke VPC Anda. Masukkan rentang dalam notasi CIDR. Sebagai contoh: 10.2.0.0/16.

Blok CIDR harus memenuhi persyaratan berikut:

- Berada dalam salah satu rentang berikut: 10.0.0.0/8, 172.16.0.0/12, atau 192.168.0.0/16.
- Memiliki ukuran minimum /24 dan ukuran maksimum /12.
- Tidak tumpang tindih dengan kisaran VPC untuk sumber daya Amazon EKS Anda.

Anda hanya dapat menentukan opsi ini saat menggunakan keluarga IPv4 alamat dan hanya pada pembuatan cluster. Jika Anda tidak menentukan ini, maka Kubernetes memberikan alamat IP layanan dari blok atau CIDR. 10.100.0.0/16 172.20.0.0/16

- Jika Anda membuat klaster dan ingin klaster menetapkan IPv6 alamat ke Pod dan layanan, bukan IPv4 alamat, tentukan opsi [IPFamily](#).

Kubernetes memberikan IPv4 alamat ke Pod dan layanan, secara default. Sebelum memutuskan untuk menggunakan IPv6 keluarga, pastikan Anda terbiasa dengan semua pertimbangan dan persyaratan dalam [the section called “Persyaratan dan pertimbangan VPC”](#), [the section called “Persyaratan dan pertimbangan subnet”](#), [the section called “Persyaratan grup keamanan”](#), dan [the section called “IPv6”](#) topik. Jika Anda memilih IPv6 keluarga, Anda tidak dapat menentukan rentang alamat untuk Kubernetes untuk menetapkan alamat IPv6 layanan seperti yang Anda bisa untuk keluarga. IPv4 Kubernetes memberikan alamat layanan dari rentang alamat lokal yang unik (). fc00::/7

Buat cluster - AWS konsol

1. Buka [konsol Amazon EKS](#).
2. Pilih Add cluster dan kemudian pilih Create.
3. Di bawah Opsi konfigurasi pilih Konfigurasi khusus
 - Untuk informasi tentang membuat cluster dengan cepat dengan Mode Otomatis EKS, lihat [the section called “Konsol manajemen”](#).
4. Di bawah Mode Otomatis EKS, matikan Gunakan Mode Otomatis EKS.
 - Untuk informasi tentang membuat kluster Mode Otomatis EKS dengan konfigurasi khusus, lihat [the section called “Buat kluster otomatis”](#).
5. Pada halaman Configure cluster, masukkan bidang-bidang berikut:
 - Nama — Nama untuk kluster Anda. Nama hanya dapat berisi karakter alfanumerik (peka huruf besar/kecil), tanda hubung, dan garis bawah. Itu harus dimulai dengan karakter alfanumerik dan tidak boleh lebih dari 100 karakter. Nama harus unik di dalam AWS Wilayah dan AWS akun tempat Anda membuat kluster.
 - Peran IAM Cluster — Pilih peran IAM kluster Amazon EKS yang Anda buat untuk memungkinkan bidang kontrol Kubernetes mengelola AWS sumber daya atas nama Anda.
 - Versi Kubernetes — Versi Kubernetes untuk digunakan untuk kluster Anda. Sebaiknya pilih versi terbaru, kecuali jika Anda memerlukan versi sebelumnya.
 - Jenis Support - Kebijakan versi Kubernetes yang ingin Anda tetapkan untuk kluster Anda. Jika Anda ingin kluster Anda hanya berjalan pada versi dukungan standar, Anda dapat memilih Dukungan standar. Jika Anda ingin kluster Anda memasukkan dukungan yang diperluas di akhir dukungan standar untuk sebuah versi, Anda dapat memilih Dukungan diperpanjang. Jika Anda memilih versi Kubernetes yang saat ini dalam dukungan diperpanjang, Anda tidak dapat memilih dukungan standar sebagai opsi.
 - Enkripsi rahasia — (Opsional) Pilih untuk mengaktifkan enkripsi rahasia Kubernetes menggunakan kunci KMS. Anda juga dapat mengaktifkan ini setelah Anda membuat cluster Anda. Sebelum Anda mengaktifkan kemampuan ini, pastikan Anda sudah terbiasa dengan informasi di dalamnya [the section called “Aktifkan enkripsi rahasia”](#).
 - Tanda — (Opsional) Tambahkan tanda apapun ke kluster Anda. Untuk informasi selengkapnya, lihat [the section called “Pemberian tag pada sumber daya Anda”](#).
 - ARC Zonal shift - (Opsional) Anda dapat menggunakan pengontrol Pemulihan Aplikasi Route53 untuk mengurangi zona ketersediaan yang terganggu. Untuk informasi selengkapnya, lihat [the section called “Pelajari tentang Zonal Shift”](#).

6. Di bagian akses Cluster pada halaman cluster konfigurasi, masukkan bidang berikut:

- Akses administrator klaster Bootstrap - Pembuat klaster secara otomatis adalah administrator Kubernetes. Jika Anda ingin menonaktifkan ini, pilih Larang akses administrator klaster.
- Mode otentikasi klaster — Tentukan cara Anda ingin memberikan akses kepada pengguna dan peran IAM ke Kubernetes. APIs Untuk informasi selengkapnya, lihat [the section called “Atur Mode Otentikasi Cluster”](#).

Setelah selesai dengan halaman ini, pilih Berikutnya.

7. Pada halaman Tentukan jaringan, pilih nilai untuk kolom berikut:

- VPC — Pilih VPC yang sudah ada yang memenuhi [persyaratan Amazon EKS VPC](#) untuk membuat klaster Anda. Sebelum memilih VPC, kami sarankan Anda memahami semua persyaratan dan pertimbangan di dalamnya. [the section called “Persyaratan VPC dan subnet”](#) Anda tidak dapat mengubah VPC mana yang ingin Anda gunakan setelah pembuatan cluster. Jika tidak VPCs terdaftar, maka Anda harus membuatnya terlebih dahulu. Untuk informasi selengkapnya, lihat [the section called “Buat VPC”](#).
- Subnet — Secara default, semua subnet yang tersedia di VPC yang ditentukan di bidang sebelumnya telah dipilih sebelumnya. Anda harus memilih setidaknya dua.

Subnet yang Anda pilih harus memenuhi persyaratan [subnet Amazon EKS](#). Sebelum memilih subnet, sebaiknya Anda terbiasa dengan semua persyaratan dan pertimbangan [VPC Amazon EKS dan subnet](#).

Grup keamanan — (Opsional) Tentukan satu atau beberapa grup keamanan yang ingin Anda kaitkan Amazon EKS ke antarmuka jaringan yang dibuatnya.

Apakah Anda memilih grup keamanan atau tidak, Amazon EKS membuat grup keamanan yang memungkinkan komunikasi antara cluster dan VPC Anda. Amazon EKS mengaitkan grup keamanan ini, dan apa pun yang Anda pilih, ke antarmuka jaringan yang dibuatnya. Untuk informasi selengkapnya tentang grup keamanan klaster yang dibuat Amazon EKS, lihat [the section called “Persyaratan grup keamanan”](#). Anda dapat mengubah aturan di grup keamanan klaster yang dibuat Amazon EKS.

- Pilih keluarga alamat IP cluster — Anda dapat memilih salah satu IPv4 dan IPv6.

Kubernetes memberikan IPv4 alamat ke Pod dan layanan, secara default. Sebelum memutuskan untuk menggunakan IPv6 keluarga, pastikan Anda sudah terbiasa dengan semua pertimbangan dan persyaratan dalam persyaratan dan pertimbangan [VPCthe section called “Persyaratan dan pertimbangan subnet”, the section called “Persyaratan grup keamanan”, dan](#)

topik. [the section called “IPv6”](#) Jika Anda memilih IPv6 keluarga, Anda tidak dapat menentukan rentang alamat untuk Kubernetes untuk menetapkan alamat IPv6 layanan seperti yang Anda bisa untuk keluarga. IPv4 Kubernetes memberikan alamat layanan dari rentang alamat lokal yang unik (). `fc00::/7`

- (Opsional) Pilih Konfigurasi rentang alamat IP Layanan Kubernetes dan tentukan rentang Layanan. **IPv4**

Menentukan jangkauan Anda sendiri dapat membantu mencegah konflik antara layanan Kubernetes dan jaringan lain yang di-peer atau terhubung ke VPC Anda. Masukkan rentang dalam notasi CIDR. Sebagai contoh: `10.2.0.0/16`.

Blok CIDR harus memenuhi persyaratan berikut:

- Berada dalam salah satu rentang berikut: `10.0.0.0/8`, `172.16.0.0/12`, atau `192.168.0.0/16`.
- Memiliki ukuran minimum `/24` dan ukuran maksimum `/12`.
- Tidak tumpang tindih dengan kisaran VPC untuk sumber daya Amazon EKS Anda.

Anda hanya dapat menentukan opsi ini saat menggunakan keluarga IPv4 alamat dan hanya pada pembuatan cluster. Jika Anda tidak menentukan ini, maka Kubernetes memberikan alamat IP layanan dari blok atau CIDR. `10.100.0.0/16` `172.20.0.0/16`

- Untuk akses endpoint Cluster, pilih opsi. Setelah cluster Anda dibuat, Anda dapat mengubah opsi ini. Sebelum memilih opsi non-default, pastikan untuk membiasakan diri dengan opsi dan implikasinya. Untuk informasi selengkapnya, lihat [the section called “Akses titik akhir kluster”](#).

Setelah selesai dengan halaman ini, pilih Berikutnya.

8. (Opsional) Pada halaman Konfigurasi observabilitas, pilih opsi pencatatan bidang Metrik dan Kontrol mana yang akan diaktifkan. Secara default, setiap jenis log dimatikan.

- Untuk informasi selengkapnya tentang opsi metrik Prometheus, lihat. [the section called “Langkah 1: Aktifkan metrik Prometheus”](#)
- Untuk informasi selengkapnya tentang opsi Pencatatan bidang kontrol, lihat [the section called “Kontrol log pesawat”](#).

Setelah selesai dengan halaman ini, pilih Berikutnya.

9. Pada halaman Pilih add-on, pilih add-on yang ingin Anda tambahkan ke cluster Anda. Add-on tertentu telah dipilih sebelumnya. Anda dapat memilih add-on Amazon EKS dan add-on AWS Marketplace sebanyak yang Anda butuhkan. Jika add-on AWS Marketplace yang ingin Anda

instal tidak terdaftar, Anda dapat mengklik penomoran halaman untuk melihat hasil halaman tambahan atau mencari add-on AWS Marketplace yang tersedia dengan memasukkan teks di kotak pencarian. Anda juga dapat memfilter berdasarkan kategori, vendor, atau model harga dan kemudian memilih add-on dari hasil pencarian. Saat membuat cluster, Anda dapat melihat, memilih, dan menginstal add-on apa pun yang mendukung EKS Pod Identities seperti yang dijelaskan di dalamnya. [the section called “Identitas Pod”](#)

Setelah selesai dengan halaman ini, pilih Berikutnya.

Beberapa add-on, seperti Amazon VPC CNI, CoreDNS, dan kube-proxy, diinstal secara default. Jika Anda menonaktifkan salah satu add-on default, ini dapat memengaruhi kemampuan Anda untuk menjalankan aplikasi Kubernetes.

- 10 Pada halaman Konfigurasi pengaturan add-on yang dipilih, pilih versi yang ingin Anda instal. Anda selalu dapat memperbarui ke versi yang lebih baru setelah pembuatan cluster.

Untuk add-on yang mendukung Identitas Pod EKS, Anda dapat menggunakan konsol untuk membuat peran secara otomatis dengan nama, kebijakan AWS terkelola, dan kebijakan kepercayaan yang telah diisi sebelumnya secara khusus untuk add-on. Anda dapat menggunakan kembali peran yang ada atau membuat peran baru untuk add-on yang didukung. Untuk langkah-langkah menggunakan konsol untuk membuat peran untuk add-on yang mendukung EKS Pod Identities, lihat. [the section called “Buat add-on \(AWS Konsol\)”](#) Jika add-on tidak mendukung EKS Pod Identity, pesan akan ditampilkan dengan instruksi untuk menggunakan wizard untuk membuat peran IAM untuk akun layanan (IRSA) setelah cluster dibuat.

Anda dapat memperbarui konfigurasi setiap add-on setelah pembuatan cluster. Untuk informasi selengkapnya tentang mengonfigurasi add-on, lihat. [the section called “Perbarui add-on”](#) Setelah selesai dengan halaman ini, pilih Berikutnya.

- 11 Pada halaman Tinjau dan buat, tinjau informasi yang Anda masukkan atau pilih pada halaman sebelumnya. Jika Anda perlu melakukan perubahan, pilih Edit. Saat Anda puas, pilih Buat. Bidang Status menunjukkan CREATING saat cluster disediakan.

Note

Anda mungkin menerima kesalahan bahwa salah satu Availability Zone dalam permintaan Anda tidak memiliki kapasitas yang cukup untuk membuat klaster Amazon EKS. Jika hal ini terjadi, output galat berisi Availability Zones yang dapat mendukung klaster baru. Cobalah untuk kembali membuat klaster dengan setidaknya dua subnet yang terletak di

Availability Zones yang didukung untuk akun Anda. Untuk informasi selengkapnya, lihat [the section called “Kapasitas tidak mencukupi”](#).

Penyediaan kluster memerlukan waktu beberapa menit.

12 Lanjutkan dengan [the section called “Langkah 3: Perbarui kubeconfig”](#)

Buat cluster - AWS CLI

1. Buat cluster Anda dengan perintah berikut. Sebelum menjalankan perintah, buat penggantian berikut:

- Ganti *region-code* dengan AWS Region tempat Anda ingin membuat cluster Anda.
- Ganti *my-cluster* dengan nama untuk cluster Anda. Nama hanya dapat berisi karakter alfanumerik (peka huruf besar/kecil), tanda hubung, dan garis bawah. Itu harus dimulai dengan karakter alfanumerik dan tidak boleh lebih dari 100 karakter. Nama harus unik di dalam AWS Wilayah dan AWS akun tempat Anda membuat kluster.
- Ganti *1.33* dengan [versi yang didukung Amazon EKS](#).
- Ganti *111122223333* dengan ID akun Anda dan *myAmazonEKSClusterRole* dengan nama peran IAM cluster Anda.
- Ganti nilainya subnetIds dengan nilai Anda sendiri. Anda juga dapat menambahkan tambahan IDs. Anda harus menentukan setidaknya dua subnet IDs.

Subnet yang Anda pilih harus memenuhi persyaratan [subnet Amazon EKS](#). Sebelum memilih subnet, sebaiknya Anda terbiasa dengan semua persyaratan dan pertimbangan [VPC Amazon EKS dan subnet](#).

- Jika Anda tidak ingin menentukan ID grup keamanan, hapus `, securityGroupIds=sg-
<ExampleID1>` dari perintah. Jika Anda ingin menentukan satu atau beberapa grup keamanan IDs, ganti nilainya securityGroupIds dengan milik Anda sendiri. Anda juga dapat menambahkan tambahan IDs.

Apakah Anda memilih grup keamanan atau tidak, Amazon EKS membuat grup keamanan yang memungkinkan komunikasi antara cluster dan VPC Anda. Amazon EKS mengaitkan grup keamanan ini, dan apa pun yang Anda pilih, ke antarmuka jaringan yang dibuatnya. Untuk informasi selengkapnya tentang grup keamanan kluster yang dibuat Amazon EKS, lihat [the](#)

[section called “Persyaratan grup keamanan”](#). Anda dapat mengubah aturan di grup keamanan klaster yang dibuat Amazon EKS.

```
aws eks create-cluster --region region-code --name my-cluster --kubernetes-version
1.33 \
  --role-arn arn:aws:iam::111122223333:role/myAmazonEKSClusterRole \
  --resources-vpc-config subnetIds=subnet-ExampleID1,subnet-
ExampleID2,securityGroupIds=sg-ExampleID1
```

Note

Anda mungkin menerima kesalahan bahwa salah satu Availability Zone dalam permintaan Anda tidak memiliki kapasitas yang cukup untuk membuat klaster Amazon EKS. Jika hal ini terjadi, output galat berisi Availability Zones yang dapat mendukung klaster baru. Cobalah untuk kembali membuat klaster dengan setidaknya dua subnet yang terletak di Availability Zones yang didukung untuk akun Anda. Untuk informasi selengkapnya, lihat [the section called “Kapasitas tidak mencukupi”](#).

Berikut ini adalah pengaturan opsional yang, jika diperlukan, harus ditambahkan ke perintah sebelumnya. Anda hanya dapat mengaktifkan opsi ini saat membuat cluster, bukan setelahnya.

- Secara default, EKS menginstal beberapa add-on jaringan selama pembuatan klaster. Ini termasuk Amazon VPC CNI, CoreDNS, dan kube-proxy.

Jika Anda ingin menonaktifkan penginstalan add-on jaringan default ini, gunakan parameter di bawah ini. Ini dapat digunakan untuk alternatif CNIs, seperti Cilium. Tinjau [referensi EKS API](#) untuk informasi lebih lanjut.

```
aws eks create-cluster --bootstrapSelfManagedAddons false
```

- Jika Anda ingin menentukan dari mana IPv4 Classless Inter-domain Routing (CIDR) blok Kubernetes memberikan alamat IP layanan, Anda harus menentukannya dengan menambahkan ke perintah berikut. `--kubernetes-network-config serviceIpv4Cidr=<cidr-block>`

Menentukan jangkauan Anda sendiri dapat membantu mencegah konflik antara layanan Kubernetes dan jaringan lain yang di-peer atau terhubung ke VPC Anda. Masukkan rentang dalam notasi CIDR. Sebagai contoh: `10.2.0.0/16`.

Blok CIDR harus memenuhi persyaratan berikut:

- Berada dalam salah satu rentang berikut: 10.0.0.0/8, 172.16.0.0/12, atau 192.168.0.0/16.
- Memiliki ukuran minimum /24 dan ukuran maksimum /12.
- Tidak tumpang tindih dengan kisaran VPC untuk sumber daya Amazon EKS Anda.

Anda hanya dapat menentukan opsi ini saat menggunakan keluarga IPv4 alamat dan hanya pada pembuatan cluster. Jika Anda tidak menentukan ini, maka Kubernetes memberikan alamat IP layanan dari blok atau CIDR. 10.100.0.0/16 172.20.0.0/16

- Jika Anda membuat klaster dan ingin klaster menetapkan IPv6 alamat ke Pod dan layanan, bukan IPv4 alamat, tambahkan `--kubernetes-network-config ipFamily=ipv6` ke perintah berikut.

Kubernetes memberikan IPv4 alamat ke Pod dan layanan, secara default. Sebelum memutuskan untuk menggunakan IPv6 keluarga, pastikan Anda sudah terbiasa dengan semua pertimbangan dan persyaratan dalam persyaratan dan pertimbangan [VPC the section called “Persyaratan dan pertimbangan subnet”](#), [the section called “Persyaratan grup keamanan”](#), dan topik. [the section called “IPv6”](#) Jika Anda memilih IPv6 keluarga, Anda tidak dapat menentukan rentang alamat untuk Kubernetes untuk menetapkan alamat IPv6 layanan seperti yang Anda bisa untuk keluarga. IPv4 Kubernetes memberikan alamat layanan dari rentang alamat lokal yang unik (). `fc00::/7`

2. Dibutuhkan beberapa menit untuk menyediakan cluster. Anda dapat melakukan kueri status klaster Anda dengan perintah berikut.

```
aws eks describe-cluster --region region-code --name my-cluster --query
"cluster.status"
```

Jangan lanjutkan ke langkah berikutnya sampai output yang dikembalikan `ACTIVE`.

3. Lanjutkan dengan [the section called “Langkah 3: Perbarui kubeconfig”](#)

Langkah 3: Perbarui kubeconfig

1. Jika Anda membuat cluster Anda menggunakan `eksctl`, maka Anda dapat melewati langkah ini. Ini karena `eksctl` sudah menyelesaikan langkah ini untuk Anda. Aktifkan `kubectl` untuk berkomunikasi dengan cluster Anda dengan menambahkan konteks baru ke `kubectl config` file. Untuk informasi selengkapnya tentang cara membuat dan memperbarui file, lihat [the section called “Akses klaster dengan kubectl”](#).


```
aws eks update-kubeconfig --region region-code --name my-cluster
```

Contoh output adalah sebagai berikut.

```
Added new context arn:aws:eks:region-code:111122223333:cluster/my-cluster to /home/username/.kube/config
```

2. Konfirmasikan komunikasi dengan cluster Anda dengan menjalankan perintah berikut.

```
kubectl get svc
```

Contoh output adalah sebagai berikut.

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	ClusterIP	10.100.0.1	<none>	443/TCP	28h

Langkah 4: Pengaturan cluster

1. (Disarankan) Untuk menggunakan beberapa add-on Amazon EKS, atau untuk mengaktifkan beban kerja Kubernetes individual agar memiliki izin AWS Identity and Access Management (IAM) tertentu, [buat penyedia IAM OpenID Connect \(OIDC\)](#) untuk klaster Anda. Anda hanya perlu membuat penyedia IAM OIDC untuk cluster Anda sekali. Untuk mempelajari selengkapnya tentang add-on Amazon EKS, lihat [the section called “Add-on Amazon EKS”](#). Untuk mempelajari selengkapnya tentang penugasan izin IAM tertentu untuk beban kerja Anda, lihat [the section called “Kredensial dengan IRSA”](#).
2. (Disarankan) Konfigurasikan klaster Anda untuk plugin Amazon VPC CNI untuk plugin Kubernetes sebelum menerapkan node Amazon ke cluster Anda. EC2 Secara default, plugin diinstal dengan cluster Anda. Saat Anda menambahkan EC2 node Amazon ke cluster Anda, plugin secara otomatis diterapkan ke setiap EC2 node Amazon yang Anda tambahkan. Plugin mengharuskan Anda untuk melampirkan salah satu kebijakan IAM berikut ke peran IAM. Jika klaster Anda menggunakan IPv4 keluarga, gunakan kebijakan IAM terkelola [Amazoneks_CNI_Policy](#). Jika klaster Anda menggunakan IPv6 keluarga, gunakan [kebijakan IAM yang Anda buat](#).

Peran IAM yang Anda lampirkan kebijakan dapat menjadi peran IAM node, atau peran khusus yang hanya digunakan untuk plugin. Kami merekomendasikan untuk melampirkan kebijakan

ke peran ini. Untuk informasi selengkapnya tentang membuat peran, lihat [the section called “Konfigurasi untuk IRSA”](#) atau [Amazon EKS node IAM role](#).

3. Jika Anda menerapkan cluster Anda menggunakan AWS Management Console, Anda dapat melewati langkah ini. Aplikasi ini AWS Management Console menerapkan plugin Amazon VPC CNI untuk add-on Kubernetes, CoreDNS, dan Amazon EKS, secara default. kube-proxy

Jika Anda menerapkan cluster Anda menggunakan salah satu eksctl atau AWS CLI, maka plugin Amazon VPC CNI untuk Kubernetes, CoreDNS, dan add-on yang dikelola sendiri akan di-deploy. kube-proxy Anda dapat memigrasikan plugin Amazon VPC CNI untuk Kubernetes, CoreDNS, dan add-on yang kube-proxy dikelola sendiri yang di-deploy dengan kluster Anda ke add-on Amazon EKS. Untuk informasi selengkapnya, lihat [the section called “Add-on Amazon EKS”](#).

4. (Opsional) Jika Anda belum melakukannya, Anda dapat mengaktifkan metrik Prometheus untuk kluster Anda. Untuk informasi selengkapnya, lihat [Membuat scraper](#) di Amazon Managed Service for Prometheus User Guide.
5. Jika Anda berencana untuk menerapkan beban kerja ke kluster yang menggunakan volume Amazon EBS, Anda harus menginstal [Amazon EBS CSI](#) ke kluster sebelum menerapkan beban kerja.

Langkah selanjutnya

- [Prinsipal IAM](#) yang menciptakan cluster adalah satu-satunya prinsipal yang memiliki akses ke cluster. [Berikan izin kepada prinsipal IAM lainnya sehingga mereka dapat mengakses kluster](#) Anda.
- Jika prinsipal IAM yang membuat kluster hanya memiliki izin IAM minimum yang direferensikan dalam prasyarat, maka Anda mungkin ingin menambahkan izin Amazon EKS tambahan untuk prinsipal tersebut. Untuk informasi selengkapnya tentang pemberian izin Amazon EKS kepada prinsipal IAM, lihat [the section called “Referensi IAM”](#)
- [Jika Anda ingin prinsipal IAM yang membuat kluster, atau prinsipal lainnya untuk melihat sumber daya Kubernetes di konsol Amazon EKS, berikan izin yang Diperlukan ke entitas.](#)
- Jika Anda ingin node dan prinsipal IAM mengakses cluster Anda dari dalam VPC Anda, aktifkan titik akhir pribadi untuk cluster Anda. Titik akhir publik diaktifkan secara default. Anda dapat menonaktifkan titik akhir publik setelah Anda mengaktifkan titik akhir pribadi, jika diinginkan. Untuk informasi selengkapnya, lihat [the section called “Akses titik akhir kluster”](#).
- [Aktifkan enkripsi rahasia untuk cluster Anda.](#)
- [Konfigurasi logging untuk kluster Anda.](#)

- [Tambahkan node ke cluster Anda.](#)

Bersiaplah untuk upgrade versi Kubernetes dan pecahkan masalah kesalahan konfigurasi dengan wawasan kluster

Wawasan kluster Amazon EKS menyediakan deteksi masalah dan rekomendasi untuk menyelesaikannya guna membantu Anda mengelola kluster. Setiap kluster Amazon EKS menjalani pemeriksaan otomatis dan berulang terhadap daftar wawasan yang dikuratori Amazon EKS. Pemeriksaan wawasan ini dikelola sepenuhnya oleh Amazon EKS dan menawarkan rekomendasi tentang cara mengatasi temuan apa pun.

Amazon EKS menyediakan dua jenis wawasan: Wawasan konfigurasi dan wawasan Peningkatan. Wawasan konfigurasi mengidentifikasi kesalahan konfigurasi dalam penyiapan Node Hibrid EKS Anda yang dapat mengganggu fungsionalitas kluster atau beban kerja Anda. Upgrade insights mengidentifikasi masalah yang dapat memengaruhi kemampuan Anda untuk meningkatkan ke versi Kubernetes yang baru.

Untuk melihat daftar pemeriksaan wawasan yang dilakukan dan masalah relevan apa pun yang telah diidentifikasi Amazon EKS, Anda dapat memanggil tampilan dalam AWS Management Console operasi, AWS CLI AWS SDKs, dan Amazon EKS ListInsights API. Untuk langkah, lihat [the section called “Lihat wawasan kluster”](#).

Frekuensi Amazon EKS menyegarkan wawasan cluster setiap 24 jam. Anda tidak dapat menyegarkan wawasan kluster secara manual. Jika Anda memperbaiki masalah kluster, perlu beberapa waktu untuk memperbarui wawasan kluster. Untuk menentukan apakah perbaikan berhasil, bandingkan waktu perubahan diterapkan dengan “waktu penyegaran terakhir” dari pemeriksaan wawasan.

Izin Amazon EKS secara otomatis membuat entri akses kluster untuk wawasan kluster di setiap kluster EKS. Entri ini memberikan izin EKS untuk melihat informasi tentang cluster Anda. EKS menggunakan informasi ini untuk menghasilkan wawasan. Untuk informasi selengkapnya, lihat [the section called “Amazon EKSCluster InsightsPolicy”](#).

Bersiaplah untuk upgrade versi Kubernetes dengan wawasan kluster

Important

Amazon EKS telah mengembalikan sementara fitur yang mengharuskan Anda menggunakan `--force` flag untuk memutakhirkan kluster Anda ketika ada masalah wawasan kluster tertentu. Untuk informasi selengkapnya, lihat [Pengembalian sementara untuk menerapkan wawasan pemutakhiran pada versi kluster pembaruan pada](#) GitHub. Untuk informasi selengkapnya tentang memperbarui kluster Anda, lihat [the section called "Langkah 3: Perbarui bidang kontrol cluster"](#).

Wawasan upgrade adalah jenis pemeriksaan wawasan tertentu dalam wawasan kluster. Pemeriksaan ini mengembalikan wawasan yang terkait dengan kesiapan peningkatan versi Kubernetes. Amazon EKS menjalankan pemeriksaan wawasan pemutakhiran pada setiap cluster EKS.

- [Sebelum memperbarui versi Kubernetes cluster Anda, periksa tab Cluster insights pada dashboard observability di konsol Amazon EKS.](#)
- Jika kluster Anda telah mengidentifikasi masalah, tinjau dan buat perbaikan yang sesuai. Masalahnya termasuk tautan ke Amazon EKS dan dokumentasi Kubernetes.
- Setelah memperbaiki masalah, tunggu hingga wawasan cluster disegarkan. Jika semua masalah telah diselesaikan, [perbarui kluster Anda](#).

Amazon EKS mengembalikan wawasan terkait kesiapan peningkatan versi Kubernetes. Wawasan upgrade mengidentifikasi kemungkinan masalah yang dapat memengaruhi peningkatan kluster Kubernetes. Ini meminimalkan upaya yang dihabiskan administrator untuk mempersiapkan peningkatan dan meningkatkan keandalan aplikasi pada versi Kubernetes yang lebih baru. Cluster secara otomatis dipindai oleh Amazon EKS terhadap daftar kemungkinan peningkatan versi Kubernetes yang berdampak pada masalah. Amazon EKS sering memperbarui daftar pemeriksaan wawasan berdasarkan ulasan perubahan yang dibuat di setiap rilis versi Kubernetes.

Wawasan peningkatan Amazon EKS mempercepat proses pengujian dan verifikasi untuk versi baru. Mereka juga memungkinkan administrator kluster dan pengembang aplikasi untuk memanfaatkan kemampuan Kubernetes terbaru dengan menyoroti masalah dan menawarkan saran remediasi.

Memecahkan masalah kesalahan konfigurasi dengan wawasan kluster

Insight kluster EKS secara otomatis memindai kluster Amazon EKS dengan node hybrid untuk mengidentifikasi masalah konfigurasi yang mengganggu plane-to-webhook komunikasi kontrol Kubernetes, perintah kubectl seperti exec dan log, dan banyak lagi. Wawasan konfigurasi memunculkan masalah dan memberikan rekomendasi remediasi, mempercepat waktu ke pengaturan node hibrida yang berfungsi penuh.

Lihat wawasan kluster

Amazon EKS menyediakan dua jenis wawasan: Wawasan konfigurasi dan wawasan Peningkatan. Wawasan konfigurasi mengidentifikasi kesalahan konfigurasi dalam penyiapan Node Hibrid EKS Anda yang dapat mengganggu fungsionalitas kluster atau beban kerja Anda. Upgrade insights mengidentifikasi masalah yang dapat memengaruhi kemampuan Anda untuk meningkatkan ke versi Kubernetes yang baru.

Untuk melihat daftar pemeriksaan wawasan yang dilakukan dan masalah relevan apa pun yang telah diidentifikasi Amazon EKS, Anda dapat memanggil tampilan dalam AWS Management Console operasi, AWS CLI AWS SDKs, dan Amazon EKS ListInsights API.

Lihat wawasan konfigurasi (Konsol)

1. Buka [konsol Amazon EKS](#).
2. Dari daftar cluster, pilih nama cluster Amazon EKS yang ingin Anda lihat wawasannya.
3. Pilih Monitor cluster.
4. Pilih tab Kesehatan Cluster.
5. Dalam tabel Configuration Insights, Anda akan melihat kolom berikut:
 - Nama — Pemeriksaan yang dilakukan oleh Amazon EKS terhadap cluster.
 - Status wawasan — Wawasan dengan status `Error` berarti ada kesalahan konfigurasi yang kemungkinan memengaruhi fungsionalitas kluster. Wawasan dengan status `Warning` berarti bahwa konfigurasi tidak cocok dengan pendekatan yang didokumentasikan, tetapi fungsionalitas kluster itu mungkin berfungsi jika Anda mengonfigurasinya dengan sengaja. Wawasan dengan status `Passing` berarti Amazon EKS tidak menemukan masalah apa pun yang terkait dengan pemeriksaan wawasan ini di cluster Anda.
 - Deskripsi — Informasi dari pemeriksaan wawasan, yang mencakup peringatan dan tindakan yang direkomendasikan untuk perbaikan.
 - Waktu penyegaran terakhir — Waktu status wawasan terakhir disegarkan untuk cluster ini.

Lihat wawasan peningkatan (Konsol)

1. Buka [konsol Amazon EKS](#).
2. Dari daftar cluster, pilih nama cluster Amazon EKS yang ingin Anda lihat wawasannya.
3. Pilih Lihat dasbor.
4. Pilih tab Cluster Insights.
5. Dalam tabel Upgrade Insights, Anda akan melihat kolom berikut:
 - Nama — Pemeriksaan yang dilakukan Amazon EKS terhadap cluster.
 - Insight status — Insight dengan status “Error” biasanya berarti versi Kubernetes yang terkena dampak adalah N+1 dari versi klaster saat ini, sedangkan status “Peringatan” berarti wawasan tersebut berlaku untuk Kubernetes versi N+2 yang akan datang atau lebih. Wawasan dengan status “Passing” berarti Amazon EKS tidak menemukan masalah apa pun yang terkait dengan pemeriksaan wawasan ini di klaster Anda. Status wawasan “Tidak Diketahui” berarti Amazon EKS tidak dapat menentukan apakah klaster Anda terpengaruh oleh pemeriksaan wawasan ini.
 - Versi — Versi Kubernetes yang diperiksa wawasan untuk kemungkinan masalah.
 - Waktu penyegaran terakhir — Waktu status wawasan terakhir disegarkan untuk cluster ini.
 - Waktu transisi terakhir — Waktu status wawasan ini terakhir berubah.
 - Deskripsi — Informasi dari pemeriksaan wawasan, yang mencakup peringatan dan tindakan yang direkomendasikan untuk perbaikan.

Lihat wawasan cluster (AWS CLI)

1. Tentukan cluster mana yang ingin Anda periksa untuk wawasan. Perintah berikut mencantumkan wawasan untuk cluster tertentu. Buat modifikasi berikut pada perintah sesuai kebutuhan dan kemudian jalankan perintah yang dimodifikasi:
 - Ganti *region-code* dengan kode untuk AWS Wilayah Anda.
 - Ganti *my-cluster* dengan nama klaster Anda.

```
aws eks list-insights --region region-code --cluster-name my-cluster
```

Contoh output adalah sebagai berikut.

```
{  
  "insights":
```

```
[
  {
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "name": "Deprecated APIs removed in Kubernetes vX.XX",
    "category": "UPGRADE_READINESS",
    "kubernetesVersion": "X.XX",
    "lastRefreshTime": 1734557315.000,
    "lastTransitionTime": 1734557309.000,
    "description": "Checks for usage of deprecated APIs that are scheduled
for removal in Kubernetes vX.XX. Upgrading your cluster before migrating to the
updated APIs supported by vX.XX could cause application impact.",
    "insightStatus":
    {
      "status": "PASSING",
      "reason": "No deprecated API usage detected within the last 30
days.",
    },
  },
  {
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "name": "Kubelet version skew",
    "category": "UPGRADE_READINESS",
    "kubernetesVersion": "X.XX",
    "lastRefreshTime": 1734557309.000,
    "lastTransitionTime": 1734557309.000,
    "description": "Checks for kubelet versions of worker nodes in the
cluster to see if upgrade would cause non compliance with supported Kubernetes
kubelet version skew policy.",
    "insightStatus":
    {
      "status": "UNKNOWN",
      "reason": "Unable to determine status of node kubelet versions.",
    },
  },
  {
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "name": "Deprecated APIs removed in Kubernetes vX.XX",
    "category": "UPGRADE_READINESS",
    "kubernetesVersion": "X.XX",
    "lastRefreshTime": 1734557315.000,
    "lastTransitionTime": 1734557309.000,
    "description": "Checks for usage of deprecated APIs that are scheduled
for removal in Kubernetes vX.XX. Upgrading your cluster before migrating to the
updated APIs supported by vX.XX could cause application impact.",
  }
]
```

```

        "insightStatus":
        {
            "status": "PASSING",
            "reason": "No deprecated API usage detected within the last 30
days.",
        },
    },
    {
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
        "name": "Cluster health issues",
        "category": "UPGRADE_READINESS",
        "kubernetesVersion": "X.XX",
        "lastRefreshTime": 1734557314.000,
        "lastTransitionTime": 1734557309.000,
        "description": "Checks for any cluster health issues that prevent
successful upgrade to the next Kubernetes version on EKS.",
        "insightStatus":
        {
            "status": "PASSING",
            "reason": "No cluster health issues detected.",
        },
    },
    {
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb",
        "name": "EKS add-on version compatibility",
        "category": "UPGRADE_READINESS",
        "kubernetesVersion": "X.XX",
        "lastRefreshTime": 1734557314.000,
        "lastTransitionTime": 1734557309.000,
        "description": "Checks version of installed EKS add-ons to ensure they
are compatible with the next version of Kubernetes. ",
        "insightStatus": { "status": "PASSING", "reason": "All installed EKS
add-on versions are compatible with next Kubernetes version." },
    },
    {
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLEccccc",
        "name": "kube-proxy version skew",
        "category": "UPGRADE_READINESS",
        "kubernetesVersion": "X.XX",
        "lastRefreshTime": 1734557314.000,
        "lastTransitionTime": 1734557309.000,
        "description": "Checks version of kube-proxy in cluster to see if
upgrade would cause non compliance with supported Kubernetes kube-proxy version
skew policy.",
    }

```



```

    "insightStatus":
    {
        "status": "PASSING",
        "reason": "kube-proxy versions match the cluster control plane
version.",
    },
    },
    {
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLEddddd",
        "name": "Deprecated APIs removed in Kubernetes vX.XX",
        "category": "UPGRADE_READINESS",
        "kubernetesVersion": "X.XX",
        "lastRefreshTime": 1734557315.000,
        "lastTransitionTime": 1734557309.000,
        "description": "Checks for usage of deprecated APIs that are scheduled
for removal in Kubernetes vX.XX. Upgrading your cluster before migrating to the
updated APIs supported by vX.XX could cause application impact.",
        "insightStatus":
        {
            "status": "PASSING",
            "reason": "No deprecated API usage detected within the last 30
days.",
        },
    },
    ],
    "nextToken": null,
}

```

2. Untuk informasi deskriptif tentang wawasan, jalankan perintah berikut. Buat modifikasi berikut pada perintah sesuai kebutuhan dan kemudian jalankan perintah yang dimodifikasi:

- Ganti *region-code* dengan kode untuk AWS Wilayah Anda.
- Ganti *a1b2c3d4-5678-90ab-cdef-EXAMPLE22222* dengan ID wawasan yang diambil dari daftar wawasan cluster.
- Ganti *my-cluster* dengan nama kluster Anda.

```
aws eks describe-insight --region region-code --id a1b2c3d4-5678-90ab-cdef-EXAMPLE22222 --cluster-name my-cluster
```

Contoh output adalah sebagai berikut.

```
{
```

```

"insight":
{
  "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
  "name": "Kubelet version skew",
  "category": "UPGRADE_READINESS",
  "kubernetesVersion": "1.27",
  "lastRefreshTime": 1734557309.000,
  "lastTransitionTime": 1734557309.000,
  "description": "Checks for kubelet versions of worker nodes in the cluster
to see if upgrade would cause non compliance with supported Kubernetes kubelet
version skew policy.",
  "insightStatus":
  {
    "status": "UNKNOWN",
    "reason": "Unable to determine status of node kubelet versions.",
  },
  "recommendation": "Upgrade your worker nodes to match the Kubernetes version
of your cluster control plane.",
  "additionalInfo":
  {
    "Kubelet version skew policy": "https://kubernetes.io/releases/version-
skew-policy/#kubelet",
    "Updating a managed node group": "https://docs.aws.amazon.com/eks/latest/
userguide/update-managed-node-group.html",
  },
  "resources": [],
  "categorySpecificSummary":
  { "deprecationDetails": [], "addonCompatibilityDetails": [] },
},
}

```

Perbarui klaster yang ada ke versi Kubernetes baru

Ketika versi Kubernetes baru tersedia di Amazon EKS, Anda dapat memperbarui kluster Amazon EKS Anda ke versi terbaru.

Important

Setelah Anda meningkatkan klaster, Anda tidak dapat menurunkan versi ke versi sebelumnya. Sebelum Anda memperbarui ke versi Kubernetes yang baru, kami sarankan

Anda meninjau informasi [the section called “Versi Kubernetes”](#) dan langkah-langkah pembaruan dalam topik ini.

Versi Kubernetes baru terkadang memperkenalkan perubahan signifikan. Oleh karena itu, kami merekomendasikan Anda menguji perilaku aplikasi Anda terhadap versi Kubernetes baru sebelum Anda memperbarui klaster produksi Anda. Anda dapat melakukannya dengan membangun alur kerja integrasi berkelanjutan untuk menguji perilaku aplikasi Anda sebelum pindah ke versi Kubernetes yang baru.

Proses update terdiri dari Amazon EKS meluncurkan simpul server API baru dengan versi Kubernetes yang diperbarui untuk menggantikan yang sudah ada. Amazon EKS melakukan infrastruktur standar dan pemeriksaan kesehatan kesiapan untuk lalu lintas jaringan pada node baru ini untuk memverifikasi bahwa mereka berfungsi seperti yang diharapkan. Namun, setelah Anda memulai upgrade cluster, Anda tidak dapat menjeda atau menghentikannya. Jika salah satu pemeriksaan ini gagal, Amazon EKS mengembalikan deployment infrastruktur itu, dan klaster Anda tetap pada versi Kubernetes sebelumnya. Menjalankan aplikasi tidak terpengaruh, dan klaster Anda tidak pernah dibiarkan dalam keadaan non-deterministik atau tidak dapat dipulihkan. Amazon EKS secara teratur mendukung semua klaster terkelola, dan mekanisme yang ada untuk memulihkan klaster jika diperlukan. Kami terus mengevaluasi dan meningkatkan proses manajemen infrastruktur Kubernetes kami.

Untuk memperbarui klaster, Amazon EKS memerlukan hingga lima alamat IP yang tersedia dari subnet yang Anda tentukan saat membuat klaster. Amazon EKS membuat antarmuka jaringan elastis klaster baru (antarmuka jaringan) di salah satu subnet yang Anda tentukan. Antarmuka jaringan dapat dibuat dalam subnet yang berbeda dari antarmuka jaringan yang ada, jadi pastikan bahwa aturan grup keamanan Anda mengizinkan [komunikasi cluster yang diperlukan](#) untuk setiap subnet yang Anda tentukan saat Anda membuat cluster Anda. Jika salah satu subnet yang Anda tentukan saat membuat klaster tidak ada, tidak memiliki cukup alamat IP yang tersedia, atau tidak memiliki aturan grup keamanan yang memungkinkan komunikasi cluster yang diperlukan, maka pembaruan dapat gagal.

Untuk memastikan bahwa endpoint server API untuk klaster Anda selalu dapat diakses, Amazon EKS menyediakan bidang kontrol Kubernetes yang sangat tersedia dan melakukan pembaruan bergulir instance server API selama operasi pembaruan. Untuk memperhitungkan perubahan alamat IP instance server API yang mendukung endpoint server API Kubernetes Anda, Anda harus memastikan bahwa klien server API Anda mengelola koneksi ulang secara efektif. Versi terbaru `kubectl` dan

[pustaka](#) klien Kubernetes yang didukung secara resmi, melakukan proses penyambungan kembali ini secara transparan.

Note

Untuk mempelajari lebih lanjut tentang apa yang masuk ke pembaruan klaster, lihat [Praktik Terbaik untuk Peningkatan Cluster](#) di Panduan Praktik Terbaik EKS. Sumber daya ini membantu Anda merencanakan peningkatan, dan memahami strategi peningkatan klaster.

Pertimbangan untuk Mode Otomatis Amazon EKS

- Kemampuan komputasi Amazon EKS Auto Mode mengontrol node versi Kubernetes. Setelah Anda memutakhirkan bidang kontrol, Mode Otomatis EKS akan mulai memperbarui node terkelola secara bertahap. Mode Otomatis EKS menghormati anggaran gangguan pod.
- Anda tidak perlu memutakhirkan kemampuan Amazon EKS Auto Mode secara manual, termasuk kemampuan penskalaan otomatis komputasi, penyimpanan blok, dan penyeimbangan beban.

Ringkasan

Ringkasan tingkat tinggi dari proses upgrade cluster Amazon EKS adalah sebagai berikut:

1. Pastikan klaster Anda dalam keadaan yang akan mendukung peningkatan. Ini termasuk memeriksa Kubernetes yang APIs digunakan oleh sumber daya yang digunakan ke dalam klaster, memastikan klaster bebas dari masalah kesehatan apa pun. Anda harus menggunakan wawasan pemutakhiran Amazon EKS saat mengevaluasi kesiapan pemutakhiran klaster Anda.
2. Tingkatkan bidang kontrol ke versi minor berikutnya (misalnya, dari 1,32 menjadi 1,33).
3. Tingkatkan node di bidang data agar sesuai dengan bidang kontrol.
4. Tingkatkan aplikasi tambahan apa pun yang berjalan di cluster (misalnya, `cluster-autoscaler`).
5. Tingkatkan add-on yang disediakan oleh Amazon EKS, seperti yang disertakan secara default:
 - [Versi yang direkomendasikan Amazon VPC CNI](#)
 - [CoreDNS versi yang direkomendasikan](#)
 - [kube-proxy versi yang direkomendasikan](#)
6. Tingkatkan setiap klien yang berkomunikasi dengan cluster (misalnya, `kubectl`).

Langkah 1: Bersiaplah untuk upgrade

Bandingkan versi Kubernetes dari pesawat kontrol klaster Anda ke versi Kubernetes simpul Anda.

- Dapatkan versi Kubernetes dari bidang kontrol cluster Anda.

```
kubectl version
```

- Dapatkan versi Kubernetes dari node Anda. Perintah ini mengembalikan semua node Amazon EC2, Fargate, dan hybrid yang dikelola sendiri dan dikelola. Setiap Pod Fargate terdaftar sebagai simpulnya sendiri.

```
kubectl get nodes
```

Sebelum memperbarui control plane Anda ke versi Kubernetes baru, pastikan bahwa versi minor Kubernetes dari kedua node terkelola dan node Fargate di cluster Anda sama dengan versi control plane Anda. Misalnya, jika bidang kontrol Anda menjalankan versi 1.29 dan salah satu node Anda menjalankan versi 1.28, maka Anda harus memperbarui node Anda ke versi 1.29 sebelum memperbarui bidang kontrol Anda ke 1.30. Kami juga menyarankan agar Anda memperbarui node yang dikelola sendiri dan node hibrida ke versi yang sama dengan bidang kontrol Anda sebelum memperbarui bidang kontrol. Lihat informasi selengkapnya di [the section called “Perbarui”](#), [the section called “Metode pembaruan”](#), dan [the section called “Tingkatkan node hibrida”](#). Jika Anda memiliki node Fargate dengan versi minor lebih rendah dari versi control plane, pertama-tama hapus Pod yang diwakili oleh node tersebut. Kemudian perbarui pesawat kontrol Anda. Pod yang tersisa akan diperbarui ke versi baru setelah Anda menerapkannya kembali.

Langkah 2: Tinjau pertimbangan peningkatan

Wawasan klaster Amazon EKS secara otomatis memindai klaster terhadap daftar potensi peningkatan versi Kubernetes yang memengaruhi masalah seperti penggunaan API Kubernetes yang tidak digunakan lagi. Amazon EKS memperbarui daftar pemeriksaan wawasan secara berkala untuk dilakukan berdasarkan evaluasi perubahan dalam proyek Kubernetes. Amazon EKS juga memperbarui daftar pemeriksaan wawasan saat perubahan diperkenalkan di layanan Amazon EKS bersama dengan versi baru. Untuk informasi selengkapnya, lihat [the section called “Wawasan cluster”](#).

Tinjau [Panduan Migrasi API yang Tidak Digunakan Lagi di dokumen Kubernetes](#).

Tinjau Wawasan Cluster

Gunakan wawasan klaster Amazon EKS untuk mengidentifikasi masalah

1. Buka [konsol Amazon EKS](#).
2. Dari daftar cluster, pilih nama cluster Amazon EKS yang ingin Anda lihat wawasannya.
3. Pilih Lihat dasbor.
4. Pilih tab Cluster Insights.
5. Dalam tabel Upgrade Insights, Anda akan melihat kolom berikut:
 - Nama — Pemeriksaan yang dilakukan oleh Amazon EKS terhadap cluster.
 - Insight status — Insight dengan status “Error” biasanya berarti versi Kubernetes yang terkena dampak adalah N+1 dari versi klaster saat ini, sedangkan status “Peringatan” berarti wawasan tersebut berlaku untuk Kubernetes versi N+2 yang akan datang atau lebih. Wawasan dengan status “Passing” berarti Amazon EKS tidak menemukan masalah apa pun yang terkait dengan pemeriksaan wawasan ini di klaster Anda. Status wawasan “Tidak Diketahui” berarti Amazon EKS tidak dapat menentukan apakah klaster Anda terpengaruh oleh pemeriksaan wawasan ini.
 - Versi — Versi Kubernetes yang diperiksa wawasan untuk kemungkinan masalah.
 - Waktu penyegaran terakhir — Waktu status wawasan terakhir disegarkan untuk cluster ini.
 - Waktu transisi terakhir — Waktu status wawasan ini terakhir berubah.
 - Deskripsi — Informasi dari pemeriksaan wawasan, yang mencakup peringatan dan tindakan yang direkomendasikan untuk perbaikan.

Pertimbangan Terperinci

- Karena Amazon EKS menjalankan bidang pengendali yang banyak tersedia, Anda dapat memperbarui hanya satu versi minor pada satu waktu. Untuk informasi selengkapnya tentang persyaratan ini, lihat [Kubernetes Version and Version Skew](#) Support Policy. Asumsikan bahwa versi cluster Anda saat ini adalah versi 1.28 dan Anda ingin memperbaruinya ke versi 1.30. Anda harus terlebih dahulu memperbarui 1.28 klaster versi Anda ke versi 1.29 dan kemudian memperbarui 1.29 klaster versi Anda ke versi 1.30.
- Tinjau versi miring antara Kubernetes kube-apiserver dan pada node Anda. kubelet
 - Mulai dari versi Kubernetes 1.28, kubelet mungkin sampai tiga versi minor lebih lama dari kube-apiserver. Lihat Kebijakan [miring versi upstream Kubernetes](#).

- Jika `kubelet` pada node terkelola dan Fargate Anda ada di versi Kubernetes 1.25 atau yang lebih baru, Anda dapat memperbarui klaster Anda hingga tiga versi ke depan tanpa memperbarui versinya. Misalnya, jika versi aktif `kubelet` 1.25, Anda dapat memperbarui versi kluster Amazon EKS Anda dari 1.25 ke 1.26, ke 1.27, dan ke 1.28 sementara versi `kubelet` tetap ada 1.25.
- Sebagai praktik terbaik sebelum memulai pembaruan, pastikan bahwa pada node Anda `kubelet` berada pada versi Kubernetes yang sama dengan bidang kontrol Anda.
- Jika cluster Anda dikonfigurasi dengan versi plugin Amazon VPC CNI untuk Kubernetes yang lebih awal dari 1.8.0, maka sebaiknya Anda memperbarui plugin ke versi terbaru sebelum memperbarui cluster Anda. Untuk memperbarui plugin, lihat [the section called “Amazon VPC CNI”](#).

Langkah 3: Perbarui bidang kontrol cluster

Important

Amazon EKS telah mengembalikan sementara fitur yang mengharuskan Anda menggunakan `--force` flag untuk memutakhirkan klaster Anda ketika ada masalah wawasan klaster tertentu. Untuk informasi selengkapnya, lihat [Pengembalian sementara untuk menerapkan wawasan pemutakhiran pada versi klaster pembaruan pada](#). GitHub

Amazon EKS menyegarkan wawasan cluster 24 jam setelah “waktu penyegaran terakhir”. Anda dapat membandingkan waktu Anda mengatasi masalah dengan “waktu penyegaran terakhir” dari wawasan cluster.

Selain itu, status insight dapat diperbaharui hingga 30 hari setelah menangani penggunaan API yang tidak digunakan lagi. Upgrade insights selalu mencari penggunaan API yang tidak digunakan lagi selama 30 hari yang bergulir.

Anda dapat mengirimkan permintaan untuk meningkatkan versi pesawat kontrol EKS Anda menggunakan:

- [eksctl](#)
- [AWS konsol](#)
- [AWS CLI](#)

Perbarui cluster - eksctl

Prosedur ini membutuhkan eksctl versi 0.210.0 atau yang lebih baru. Anda dapat memeriksa versi Anda dengan perintah berikut:

```
eksctl version
```

Untuk petunjuk tentang cara menginstal dan memperbaruieksctl, lihat [Instalasi](#) dalam eksctl dokumentasi.

Perbarui versi Kubernetes dari pesawat kontrol Amazon EKS Anda. Ganti <cluster-name> dengan nama klaster Anda. Ganti <version-number> dengan nomor versi yang didukung Amazon EKS yang ingin Anda perbarui klaster Anda. Untuk daftar nomor versi yang didukung, lihat [the section called “Versi Kubernetes”](#).

```
eksctl upgrade cluster --name <cluster-name> --version <version-number> --approve
```

Pembaruan memerlukan waktu beberapa menit.

Lanjutkan ke [the section called “Langkah 4: Perbarui komponen cluster”](#).

Perbarui cluster - AWS konsol

1. Buka [konsol Amazon EKS](#).
2. Pilih Upgrade sekarang untuk cluster yang ingin Anda upgrade.
3. Pilih versi untuk memperbarui cluster Anda dan pilih Upgrade.
4. Pembaruan memerlukan waktu beberapa menit. Lanjutkan ke [the section called “Langkah 4: Perbarui komponen cluster”](#).

Perbarui cluster - AWS CLI

1. Verifikasi bahwa AWS CLI diinstal dan Anda masuk. Untuk informasi selengkapnya, lihat [Menginstal atau memperbarui ke versi terbaru AWS CLI](#).
2. Perbarui cluster Amazon EKS Anda dengan perintah AWS CLI berikut. Ganti <cluster-name> dan <region-code> cluster yang ingin Anda tingkatkan. Ganti <version-number> dengan nomor versi yang didukung Amazon EKS yang ingin Anda perbarui klaster Anda. Untuk daftar nomor versi yang didukung, lihat [the section called “Versi Kubernetes”](#).


```
aws eks update-cluster-version --name <cluster-name> \
  --kubernetes-version <version-number> --region <region-code>
```

Contoh output adalah sebagai berikut.

```
{
  "update": {
    "id": "<update-id>",
    "status": "InProgress",
    "type": "VersionUpdate",
    "params": [
      {
        "type": "Version",
        "value": "<version-number>"
      },
      {
        "type": "PlatformVersion",
        "value": "eks.1"
      }
    ],
    [...]
    "errors": []
  }
}
```

3. Pembaruan memerlukan waktu beberapa menit. Pantau status pembaruan klaster Anda dengan perintah berikut. Selain menggunakan yang sama <cluster-name> dan <region-code>, gunakan <update-id> yang dikembalikan oleh perintah sebelumnya.

```
aws eks describe-update --name <cluster-name> \
  --region <region-code> --update-id <update-id>
```

Ketika Successful status ditampilkan, pembaruan selesai.

4. Lanjutkan ke [the section called “Langkah 4: Perbarui komponen cluster”](#).

Langkah 4: Perbarui komponen cluster

1. Setelah pembaruan klaster Anda selesai, perbarui node Anda ke versi minor Kubernetes yang sama dengan klaster Anda yang diperbarui. Lihat informasi selengkapnya di [the section called “Metode pembaruan”](#), [the section called “Perbarui”](#), dan [the section called “Tingkatkan node”](#)

[hibrida](#)". Pod baru apa pun yang diluncurkan di Fargate memiliki kubelet versi yang cocok dengan versi klaster Anda. Pod Fargate yang ada tidak diubah.

2. (Opsional) Jika Anda men-deploy Kubernetes Cluster Autoscaler ke klaster Anda sebelum memperbarui klaster, perbarui Cluster Autoscaler ke versi terbaru yang cocok dengan Kubernetes versi major dan minor yang Anda perbarui.
 - a. Buka halaman [rilis](#) Cluster Autoscaler di browser web dan temukan versi Cluster Autoscaler terbaru yang cocok dengan versi mayor dan minor Kubernetes klaster Anda. Misalnya, jika versi Kubernetes klaster Anda 1.30 menemukan rilis Cluster Autoscaler terbaru yang dimulai dengan 1.30 Catat nomor versi semantik (1.30.n, misalnya) untuk rilis yang akan digunakan pada langkah berikutnya.
 - b. Atur tanda citra Cluster Autoscaler ke versi yang Anda catat di langkah sebelumnya dengan perintah berikut. Jika perlu, ganti X.XX.X dengan nilai Anda sendiri.

```
kubectl -n kube-system set image deployment.apps/cluster-autoscaler cluster-autoscaler=registry.k8s.io/autoscaling/cluster-autoscaler:vX.XX.X
```

3. (Cluster dengan node GPU saja) Jika cluster Anda memiliki grup node dengan dukungan GPU (misalnya, p3.2xlarge), Anda harus memperbarui [plugin perangkat NVIDIA untuk DaemonSet Kubernetes](#) di cluster Anda. Ganti <vX.X.X> dengan s-device-plugin versi [NVIDIA/K8](#) yang Anda inginkan sebelum menjalankan perintah berikut.

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/<vX.X.X>/deployments/static/nvidia-device-plugin.yml
```

4. Perbarui plugin Amazon VPC CNI untuk Kubernetes, CoreDNS, dan add-on. kube-proxy Kami menyarankan untuk memperbarui add-on ke versi minimum yang tercantum dalam [token akun Layanan](#).
 - Jika Anda menggunakan add-on Amazon EKS, pilih Cluster di konsol Amazon EKS, lalu pilih nama cluster yang Anda perbarui di panel navigasi kiri. Pemberitahuan muncul di konsol. Mereka memberi tahu Anda bahwa versi baru tersedia untuk setiap add-on yang memiliki pembaruan yang tersedia. Untuk memperbarui add-on, pilih tab Add-ons. Di salah satu kotak untuk add-on yang memiliki pembaruan yang tersedia, pilih Perbarui sekarang, pilih versi yang tersedia, lalu pilih Perbarui.
 - Sebagai alternatif, Anda dapat menggunakan AWS CLI atau eksctl memperbarui add-on. Untuk informasi selengkapnya, lihat [the section called "Perbarui add-on"](#).
5. Jika perlu, perbarui versi Anda kubectl. Anda harus menggunakan versi kubectl yang tidak lebih dari satu perbedaan kecil dari bidang kendali Amazon EKS klaster Anda.

Turunkan versi Kubernetes untuk kluster Amazon EKS

Anda tidak dapat menurunkan versi Kubernetes dari cluster Amazon EKS. Sebagai gantinya, buat cluster baru pada versi Amazon EKS sebelumnya dan memigrasikan beban kerja.

Hapus kluster

Setelah selesai menggunakan kluster Amazon EKS, Anda harus menghapus sumber daya yang terkait dengannya sehingga Anda tidak dikenakan biaya yang tidak perlu.

Anda dapat menghapus cluster dengan `kubectl`, AWS Management Console, atau AWS CLI.

Pertimbangan

- Jika Anda memiliki layanan aktif di kluster yang terkait dengan penyeimbang beban, Anda harus menghapus layanan tersebut sebelum menghapus kluster agar penyeimbang beban terhapus dengan benar. Jika tidak, Anda dapat memiliki sumber daya orphaned di VPC yang mencegah Anda untuk dapat menghapus VPC.
- Jika Anda menerima kesalahan karena pembuat kluster telah dihapus, lihat [artikel ini](#) untuk menyelesaikannya.
- Layanan Terkelola Amazon untuk sumber daya Prometheus berada di luar siklus hidup kluster dan perlu dipertahankan secara independen dari kluster. Saat Anda menghapus kluster Anda, pastikan juga untuk menghapus pencakar yang berlaku untuk menghentikan biaya yang berlaku. Untuk informasi selengkapnya, lihat [Menemukan dan menghapus pencakar](#) di Amazon Managed Service for Prometheus User Guide.
- Untuk menghapus kluster yang terhubung, lihat [the section called “Deregister sebuah cluster”](#)

Pertimbangan untuk Mode Otomatis EKS

- Setiap Node Mode Otomatis EKS akan dihapus, termasuk instans yang EC2 dikelola
- Semua load balancer akan dihapus

Untuk informasi selengkapnya, lihat [the section called “Nonaktifkan Mode Otomatis EKS”](#).

Hapus cluster (eksctl)

Prosedur ini membutuhkan eksctl versi 0.210.0 atau yang lebih baru. Anda dapat memeriksa versi Anda dengan perintah berikut:

```
eksctl version
```

Untuk petunjuk tentang cara menginstal atau meningkatkan eksctl, lihat [Instalasi](#) dalam eksctl dokumentasi.

1. Buat daftar semua layanan yang berjalan di klaster Anda.

```
kubectl get svc --all-namespaces
```

- a. Hapus layanan apa pun yang memiliki nilai EXTERNAL-IP yang terkait. Layanan-layanan ini didukung oleh penyeimbang beban Elastic Load Balancing, dan Anda harus menghapusnya di Kubernetes agar penyeimbang beban dan sumber daya terkait dilepaskan dengan benar. Ganti *service-name* dengan nama setiap layanan yang tercantum seperti yang dijelaskan.

```
kubectl delete svc service-name
```

2. Hapus cluster dan node yang terkait dengan perintah berikut, ganti *prod* dengan nama cluster Anda.

```
eksctl delete cluster --name prod
```

Output:

```
[#] using region region-code
[#] deleting EKS cluster "prod"
[#] will delete stack "eksctl-prod-nodegroup-standard-nodes"
[#] waiting for stack "eksctl-prod-nodegroup-standard-nodes" to get deleted
[#] will delete stack "eksctl-prod-cluster"
[#] the following EKS cluster resource(s) for "prod" will be deleted: cluster. If in
doubt, check CloudFormation console
```

Hapus cluster (AWS konsol)

1. Buat daftar semua layanan yang berjalan di klaster Anda.

```
kubectl get svc --all-namespaces
```

2. Hapus layanan apa pun yang memiliki nilai EXTERNAL-IP yang terkait. Layanan-layanan ini didukung oleh penyeimbang beban Elastic Load Balancing, dan Anda harus menghapusnya di Kubernetes agar penyeimbang beban dan sumber daya terkait dilepaskan dengan benar. Ganti *service-name* dengan nama setiap layanan yang tercantum seperti yang dijelaskan.

```
kubectl delete svc service-name
```

3. Hapus semua grup simpul dan profil Fargate.
 - a. Buka [konsol Amazon EKS](#).
 - b. Di panel navigasi kiri, pilih Amazon EKS Clusters, lalu di daftar tab cluster, pilih nama cluster yang ingin Anda hapus.
 - c. Pilih tab Compute dan pilih grup node untuk dihapus. Pilih Hapus, masukkan nama grup simpul, lalu pilih Hapus. Hapus semua grup simpul dalam klaster.

Note

Grup-grup simpul yang terdaftar hanyalah [grup simpul dikelola](#).

- d. Pilih Profil Fargate untuk dihapus, pilih Hapus, masukkan nama profil, lalu pilih Hapus. Hapus semua profil Fargate di klaster.
4. Hapus semua AWS CloudFormation tumpukan node yang dikelola sendiri.
 - a. Buka [konsol AWS CloudFormation](#).
 - b. Pilih tumpukan simpul yang akan dihapus, lalu pilih Hapus.
 - c. Di kotak dialog Hapus tumpukan konfirmasi, pilih Hapus tumpukan. Hapus semua tumpukan simpul swakelola di dalam klaster.
 5. Hapus klaster .
 - a. Buka [konsol Amazon EKS](#).
 - b. pilih cluster yang akan dihapus dan pilih Hapus.
 - c. Pada layar konfirmasi hapus klaster, pilih Hapus.
 6. (Opsional) Hapus tumpukan VPC AWS CloudFormation .

- a. Buka [konsol AWS CloudFormation](#) .
- b. Pilih tumpukan VPC yang akan dihapus, lalu pilih Hapus.
- c. Di kotak dialog Hapus tumpukan konfirmasi, pilih Hapus tumpukan.

Hapus cluster (AWS CLI)

1. Buat daftar semua layanan yang berjalan di kluster Anda.

```
kubectl get svc --all-namespaces
```

2. Hapus layanan apa pun yang memiliki nilai EXTERNAL - IP yang terkait. Layanan-layanan ini didukung oleh penyeimbang beban Elastic Load Balancing, dan Anda harus menghapusnya di Kubernetes agar penyeimbang beban dan sumber daya terkait dilepaskan dengan benar. Ganti *service-name* dengan nama setiap layanan yang tercantum seperti yang dijelaskan.

```
kubectl delete svc service-name
```

3. Hapus semua grup simpul dan profil Fargate.

- a. Buat daftar grup simpul di kluster Anda dengan perintah berikut.

```
aws eks list-nodegroups --cluster-name my-cluster
```



Note

Grup-grup simpul yang terdaftar adalah [grup simpul dikelola](#) saja.

- b. Hapus setiap grup simpul dengan perintah berikut. Hapus semua grup simpul dalam kluster.

```
aws eks delete-nodegroup --nodegroup-name my-nodegroup --cluster-name my-cluster
```

- c. Buat daftar profil Fargate di kluster Anda dengan perintah berikut.

```
aws eks list-fargate-profiles --cluster-name my-cluster
```

- d. Hapus setiap profil Fargate dengan perintah berikut. Hapus semua profil Fargate di dalam kluster.

```
aws eks delete-fargate-profile --fargate-profile-name my-fargate-profile --  
cluster-name my-cluster
```

4. Hapus semua AWS CloudFormation tumpukan node yang dikelola sendiri.

- a. Buat daftar AWS CloudFormation tumpukan yang tersedia dengan perintah berikut. Temukan nama templat simpul dalam output yang dihasilkan.

```
aws cloudformation list-stacks --query "StackSummaries[].StackName"
```

- b. Hapus setiap tumpukan node dengan perintah berikut, ganti *node-stack* dengan nama tumpukan node Anda. Hapus semua tumpukan simpul swakelola di dalam kluster.

```
aws cloudformation delete-stack --stack-name node-stack
```

5. Hapus cluster dengan perintah berikut, ganti *my-cluster* dengan nama cluster Anda.

```
aws eks delete-cluster --name my-cluster
```

6. (Opsional) Hapus tumpukan VPC AWS CloudFormation .

- a. Buat daftar AWS CloudFormation tumpukan yang tersedia dengan perintah berikut. Temukan nama templat VPC dalam output yang dihasilkan.

```
aws cloudformation list-stacks --query "StackSummaries[].StackName"
```

- b. Hapus tumpukan VPC dengan perintah berikut, ganti *my-vpc-stack* dengan nama tumpukan VPC Anda.

```
aws cloudformation delete-stack --stack-name my-vpc-stack
```

Titik akhir server API cluster

Topik ini membantu Anda mengaktifkan akses pribadi untuk titik akhir server API Kubernetes kluster Amazon EKS Anda dan membatasi, atau sepenuhnya menonaktifkan, akses publik dari internet.

Ketika Anda membuat kluster baru, Amazon EKS membuat titik akhir untuk server API Kubernetes terkelola yang Anda gunakan untuk berkomunikasi dengan kluster Anda (alat pengelolaan Kubernetes seperti `kubectl`). Secara default, endpoint server API ini bersifat publik ke internet, dan

akses ke server API diamankan menggunakan kombinasi AWS Identity and Access Management (IAM) and Access Management (IAM) dan native [Kubernetes](#) Role Based Access Control (RBAC). Endpoint ini dikenal sebagai cluster public endpoint. Juga ada titik akhir pribadi cluster. Untuk informasi selengkapnya tentang titik akhir pribadi cluster, lihat bagian [the section called “Titik akhir pribadi cluster”](#) berikut.

IPv6format titik akhir cluster

EKS membuat titik akhir dual-stack unik dalam format berikut untuk IPv6 cluster baru yang dibuat setelah Oktober 2024. Cluster adalah IPv6 cluster yang Anda pilih IPv6 dalam pengaturan IP family (ipFamily) cluster.

Example

AWS

Titik akhir publik/pribadi kluster EKS: `eks-cluster.region.api.aws`

AWS GovCloud (US)

Titik akhir publik/pribadi kluster EKS: `eks-cluster.region.api.aws`

Amazon Web Services in China

Titik akhir publik/pribadi kluster EKS: `eks-cluster.region.api.amazonwebservices.com.cn`

Note

Titik akhir cluster dual-stack diperkenalkan pada Oktober 2024. Untuk informasi selengkapnya tentang IPv6 cluster, lihat [the section called “IPv6”](#). Cluster yang dibuat sebelum Oktober 2024, gunakan format titik akhir berikut sebagai gantinya.

IPv4format titik akhir cluster

EKS membuat titik akhir unik dalam format berikut untuk setiap cluster yang memilih IPv4 dalam pengaturan keluarga IP (IPFamily) cluster:

Example

AWS

EKS cluster titik akhir publik/pribadi `eks-cluster.region.eks.amazonaws.com`

AWS GovCloud (US)

EKS cluster titik akhir publik/pribadi `eks-cluster.region.eks.amazonaws.com`

Amazon Web Services in China

EKS cluster titik akhir publik/pribadi `eks-cluster.region.amazonwebservices.com.cn`

Note

Sebelum Oktober 2024, IPv6 cluster menggunakan format titik akhir ini juga. Untuk cluster tersebut, titik akhir publik dan titik akhir pribadi hanya memiliki IPv4 alamat penyelesaian dari titik akhir ini.

Titik akhir pribadi cluster

Anda dapat mengaktifkan akses privat ke server API Kubernetes sehingga semua komunikasi antara simpul Anda dan server API tetap berada di dalam VPC Anda. Anda dapat membatasi alamat IP yang dapat mengakses server API Anda dari internet, atau menonaktifkan sepenuhnya akses internet ke server API.

Note

Karena titik akhir ini untuk server API Kubernetes dan bukan AWS PrivateLink titik akhir tradisional untuk berkomunikasi dengan AWS API, titik akhir ini tidak muncul sebagai titik akhir di konsol VPC Amazon.

Saat Anda mengaktifkan akses pribadi titik akhir untuk klaster Anda, Amazon EKS membuat zona host pribadi Route 53 atas nama Anda dan mengaitkannya dengan VPC klaster Anda. Zona host pribadi ini dikelola oleh Amazon EKS, dan tidak muncul di sumber daya Route 53 akun Anda. Agar zona privat yang di-hosting dapat merutekan lalu lintas ke server API Anda dengan benar, VPC Anda

harus memiliki `enableDnsHostnames` dan `enableDnsSupport` yang diatur ke `true`, dan opsi DHCP yang diatur untuk VPC Anda harus menyertakan `AmazonProvidedDNS` dalam daftar server nama domainnya. Untuk informasi selengkapnya, lihat [Memperbarui DNS dukungan untuk VPC Anda](#) dalam Panduan Pengguna Amazon VPC.

Anda dapat menentukan persyaratan akses titik akhir server API ketika membuat kluster baru, dan dapat memperbarui akses tersebut untuk kluster kapanpun.

Memodifikasi akses titik akhir kluster

Gunakan prosedur di bagian ini untuk memodifikasi akses titik akhir untuk kluster yang sudah ada. Tabel berikut menunjukkan kombinasi akses titik akhir server API yang didukung dan perilaku terkaitnya.

Akses publik titik akhir	Akses privat titik akhir	Perilaku
Diaktifkan	Dinonaktifkan	- Ini adalah perilaku default untuk kluster Amazon EKS yang baru. - Permintaan API Kubernetes yang berasal dari dalam VPC kluster Anda (seperti node untuk mengontrol komunikasi pesawat) meninggalkan VPC tetapi bukan jaringan Amazon. - Server API kluster Anda dapat diakses dari internet. Secara opsional, Anda dapat membatasi blok CIDR yang dapat mengakses titik akhir publik. Jika Anda membatasi akses ke blok CIDR tertentu, disarankan agar Anda juga mengaktifkan titik akhir pribadi, atau memastikan bahwa blok CIDR yang Anda tentukan menyertakan alamat tempat node dan Pod Fargate (jika Anda menggunakannya) mengakses titik akhir publik.
Diaktifkan	Diaktifkan	Permintaan API Kubernetes dalam VPC kluster Anda (seperti node untuk mengontrol komunikasi pesawat) menggunakan titik akhir VPC pribadi.

Akses publik titik akhir	Akses privat titik akhir	Perilaku
		• Server API klaster Anda dapat diakses dari internet. Secara opsional, Anda dapat membatasi blok CIDR yang dapat mengakses titik akhir publik. • Jika Anda menggunakan node hibrida dengan kluster Amazon EKS Anda, tidak disarankan untuk mengaktifkan akses titik akhir klaster Publik dan Pribadi. Karena node hybrid Anda berjalan di luar VPC Anda, mereka akan menyelesaikan titik akhir cluster ke alamat IP publik. Disarankan untuk menggunakan akses endpoint klaster Publik atau Pribadi untuk cluster dengan node hibrida.
Nonaktif	Diaktifkan	• Semua lalu lintas ke server API cluster Anda harus berasal dari dalam VPC klaster Anda atau jaringan yang terhubung. • Tidak ada akses publik ke server API Anda dari internet. Perintah <code>kubectl</code> apa pun harus berasal dari dalam VPC atau jaringan yang terhubung. Untuk opsi koneksi lainnya, lihat the section called “Mengakses server API privat saja”. • Titik akhir server API cluster diselesaikan oleh server DNS publik ke alamat IP pribadi dari VPC. Di masa lalu, titik akhir hanya dapat diselesaikan dari dalam VPC. Jika titik akhir Anda tidak menyelesaikan ke alamat IP privat dalam VPC untuk klaster yang sudah ada, Anda dapat: • Aktifkan akses publik dan kemudian nonaktifkan kembali. Anda hanya perlu melakukannya sekali untuk sebuah klaster dan titik akhir akan diselesaikan ke alamat IP privat sejak saat itu. • Perbarui klaster Anda.

Blok CIDR di titik akhir publik (cluster) **IPv6**

Anda dapat menambahkan IPv6 dan IPv4 CIDR memblokir ke titik akhir publik sebuah IPv6 cluster, karena titik akhir publik adalah dual-stack. Ini hanya berlaku untuk cluster baru dengan ipFamily set IPv6 yang Anda buat pada Oktober 2024 atau lebih baru. Anda dapat mengidentifikasi cluster ini dengan nama domain endpoint baru. `api.aws`

Blok CIDR di titik akhir publik (cluster) **IPv4**

Anda dapat menambahkan blok IPv4 CIDR ke titik akhir publik sebuah IPv4 cluster. Anda tidak dapat menambahkan blok IPv6 CIDR ke titik akhir publik klaster. IPv4 Jika Anda mencoba, EKS mengembalikan pesan kesalahan berikut: `The following CIDRs are invalid in publicAccessCidrs`

Blok CIDR di titik akhir publik (**IPv6**cluster dibuat sebelum Oktober 2024)

Anda dapat menambahkan blok IPv4 CIDR ke titik akhir publik dari IPv6 cluster lama yang Anda buat sebelum Oktober 2024. Anda dapat mengidentifikasi cluster ini dengan titik `eks.amazonaws.com` akhir. Anda tidak dapat menambahkan blok IPv6 CIDR ke titik akhir publik dari IPv6 cluster lama yang Anda buat sebelum Oktober 2024. Jika Anda mencoba, EKS mengembalikan pesan kesalahan berikut: `The following CIDRs are invalid in publicAccessCidrs`

Mengakses server API privat saja

[Jika Anda telah menonaktifkan akses publik untuk titik akhir server Kubernetes API klaster Anda, Anda hanya dapat mengakses server API dari dalam VPC atau jaringan yang terhubung.](#) Berikut adalah beberapa alternatif cara untuk mengakses titik akhir server Kubernetes API:

Jaringan yang terhubung

Hubungkan jaringan Anda ke VPC dengan [gateway AWS transit](#) atau opsi [konektivitas](#) lainnya dan kemudian gunakan komputer di jaringan yang terhubung. Anda harus memastikan bahwa grup keamanan pesawat kendali Amazon EKS Anda berisi aturan yang mengizinkan lalu lintas masuk di port 443 dari jaringan terhubung milik Anda.

Tuan rumah EC2 benteng Amazon

Anda dapat meluncurkan EC2 instans Amazon ke subnet publik di VPC klaster Anda dan kemudian masuk melalui SSH ke instance tersebut untuk menjalankan perintah. `kubectl` Untuk

informasi selengkapnya, lihat [Linux bastion host di AWS](#). Anda harus memastikan bahwa grup keamanan pesawat kendali Amazon EKS Anda berisi aturan yang mengizinkan lalu lintas masuk di port 443 dari host bastion milik Anda. Untuk informasi selengkapnya, lihat [the section called “Persyaratan grup keamanan”](#).

Saat Anda mengonfigurasi `kubectl` untuk host bastion Anda, pastikan untuk menggunakan AWS kredensial yang sudah dipetakan ke konfigurasi RBAC cluster Anda, atau tambahkan [prinsipal IAM](#) yang akan digunakan bastion Anda ke konfigurasi RBAC sebelum Anda menghapus akses publik titik akhir. Untuk informasi selengkapnya, lihat [the section called “Akses API Kubernetes”](#) dan [the section called “Tidak sah atau akses ditolak \(kubectl\)”](#).

AWS Cloud9 IDE

AWS Cloud9 adalah lingkungan pengembangan terintegrasi berbasis cloud (IDE) yang memungkinkan Anda menulis, menjalankan, dan men-debug kode hanya dengan browser. Anda dapat membuat AWS Cloud9 IDE di VPC klaster Anda dan menggunakan IDE untuk berkomunikasi dengan cluster Anda. Untuk informasi selengkapnya, lihat [Membuat lingkungan di AWS Cloud9](#). Anda harus memastikan bahwa grup keamanan pesawat kendali Amazon EKS Anda berisi aturan yang mengizinkan lalu lintas masuk di port 443 dari grup keamanan IDE milik Anda. Untuk informasi selengkapnya, lihat [the section called “Persyaratan grup keamanan”](#).

Saat Anda mengonfigurasi `kubectl` untuk AWS Cloud9 IDE, pastikan untuk AWS menggunakan kredensial yang sudah dipetakan ke konfigurasi RBAC klaster Anda, atau tambahkan prinsip IAM yang akan digunakan IDE Anda ke konfigurasi RBAC sebelum Anda menghapus akses publik titik akhir. Untuk informasi selengkapnya, lihat [the section called “Akses API Kubernetes”](#) dan [the section called “Tidak sah atau akses ditolak \(kubectl\)”](#).

[Edit halaman ini di GitHub](#)

Konfigurasi akses jaringan ke titik akhir server API cluster

Anda dapat memodifikasi akses endpoint server API cluster menggunakan AWS Management Console atau AWS CLI di bagian berikut.

Konfigurasi akses titik akhir - konsol AWS

1. Buka [konsol Amazon EKS](#).
2. Pilih nama klaster untuk menampilkan informasi klaster Anda.
3. Pilih tab Jaringan dan pilih Kelola akses titik akhir.

4. Untuk akses Pribadi, pilih apakah akan mengaktifkan atau menonaktifkan akses pribadi untuk titik akhir server Kubernetes API klaster Anda. Jika Anda mengaktifkan akses pribadi, permintaan Kubernetes API yang berasal dari dalam VPC klaster Anda menggunakan titik akhir VPC pribadi. Anda harus mengaktifkan akses privat untuk menonaktifkan akses publik.
5. Untuk akses Publik, pilih apakah akan mengaktifkan atau menonaktifkan akses publik untuk titik akhir server Kubernetes API klaster Anda. Jika Anda menonaktifkan akses publik, server Kubernetes API klaster Anda hanya dapat menerima permintaan dari dalam VPC klaster.
6. (Opsional) Jika Anda telah mengaktifkan akses Publik, Anda dapat menentukan alamat mana dari internet yang dapat berkomunikasi ke titik akhir publik. Pilih Pengaturan lanjutan. Masukkan blok CIDR, seperti `203.0.113.5/32`. Blok tidak dapat mencakup [reserved addresses](#). Anda dapat memasukkan blok tambahan dengan memilih Tambahkan sumber. Ada jumlah maksimum blok CIDR yang dapat Anda tentukan. Untuk informasi selengkapnya, lihat [the section called “Kuota layanan”](#). Jika Anda menetapkan tidak ada blok, maka titik akhir server API publik menerima permintaan dari semua alamat IP untuk keduanya IPv4 (`0.0.0.0/0`) dan tambahan IPv6 (`::/0`) untuk cluster dual-stack IPv6. Jika Anda membatasi akses ke titik akhir publik Anda menggunakan blok CIDR, kami sarankan Anda juga mengaktifkan akses endpoint pribadi sehingga node dan Fargate Pods (jika Anda menggunakannya) dapat berkomunikasi dengan cluster. Tanpa mengaktifkan titik akhir privat, sumber CIDR titik akhir akses publik Anda harus menyertakan sumber keluar dari VPC Anda. Misalnya, jika Anda memiliki simpul di subnet privat yang berkomunikasi dengan internet melalui NAT Gateway, Anda perlu menambahkan alamat IP keluar dari NAT Gateway sebagai bagian dari blok CIDR yang diizinkan di titik akhir publik Anda.
7. Pilih Perbarui untuk menyelesaikan.

Konfigurasi akses titik akhir - AWS CLI

Selesaikan langkah-langkah berikut menggunakan versi AWS CLI 1.27.160 atau yang lebih baru. Anda dapat memeriksa versi saat ini dengan `aws --version`. Untuk menginstal atau meningkatkan AWS CLI, lihat [Menginstal CLI AWS](#).

1. Perbarui akses endpoint server API cluster Anda dengan perintah AWS CLI berikut. Gantikan nama klaster Anda dan nilai akses titik akhir yang diinginkan. Jika Anda mengatur `endpointPublicAccess=true`, maka Anda dapat (opsional) memasukkan satu blok CIDR, atau daftar blok CIDR yang dipisahkan koma untuk `publicAccessCidrs`. Blok tidak dapat mencakup [alamat yang disimpan](#). Jika Anda menentukan blok CIDR, maka titik akhir server API publik hanya akan menerima permintaan dari blok yang terdaftar. Ada jumlah maksimum blok CIDR yang dapat Anda tentukan. Untuk informasi selengkapnya, lihat [the section called “Kuota layanan”](#).

Jika Anda membatasi akses ke titik akhir publik Anda menggunakan blok CIDR, disarankan agar Anda juga mengaktifkan akses endpoint pribadi sehingga node dan Fargate Pods (jika Anda menggunakannya) dapat berkomunikasi dengan cluster. Tanpa mengaktifkan titik akhir privat, sumber CIDR titik akhir akses publik Anda harus menyertakan sumber keluar dari VPC Anda. Misalnya, jika Anda memiliki simpul di subnet privat yang berkomunikasi dengan internet melalui NAT Gateway, Anda perlu menambahkan alamat IP keluar dari NAT Gateway sebagai bagian dari blok CIDR yang diizinkan di titik akhir publik Anda. Jika Anda menentukan tidak ada blok CIDR, maka titik akhir server API publik menerima permintaan dari semua alamat IP (0.0.0.0/0) dan tambahan IPv6 () untuk cluster dual-stack. `::/0` IPv6

 Note

Perintah berikut memungkinkan akses privat dan akses publik dari satu alamat IP untuk titik akhir server API. Ganti `203.0.113.5/32` dengan satu blok CIDR, atau daftar blok CIDR yang dipisahkan koma yang ingin Anda batasi akses jaringan.

```
aws eks update-cluster-config \
  --region region-code \
  --name my-cluster \
  --resources-vpc-config
endpointPublicAccess=true,publicAccessCidrs="203.0.113.5/32",endpointPrivateAccess=true
```

Contoh output adalah sebagai berikut.

```
{
  "update": {
    "id": "e6f0905f-a5d4-4a2a-8c49-EXAMPLE000000",
    "status": "InProgress",
    "type": "EndpointAccessUpdate",
    "params": [
      {
        "type": "EndpointPublicAccess",
        "value": "true"
      },
      {
        "type": "EndpointPrivateAccess",
        "value": "true"
      }
    ]
  }
}
```

```

        "type": "publicAccessCidrs",
        "value": "[\"203.0.113.5/32\"]"
      }
    ],
    "createdAt": 1576874258.137,
    "errors": []
  }
}

```

2. Pantau status pembaruan akses titik akhir Anda dengan perintah berikut, menggunakan nama kluster dan ID pembaruan yang dikembalikan oleh perintah sebelumnya. Pembaruan Anda selesai saat status ditampilkan sebagai `Successful`.

```

aws eks describe-update \
  --region region-code \
  --name my-cluster \
  --update-id e6f0905f-a5d4-4a2a-8c49-EXAMPLE000000

```

Contoh output adalah sebagai berikut.

```

{
  "update": {
    "id": "e6f0905f-a5d4-4a2a-8c49-EXAMPLE000000",
    "status": "Successful",
    "type": "EndpointAccessUpdate",
    "params": [
      {
        "type": "EndpointPublicAccess",
        "value": "true"
      },
      {
        "type": "EndpointPrivateAccess",
        "value": "true"
      },
      {
        "type": "publicAccessCidrs",
        "value": "[\"203.0.113.5/32\"]"
      }
    ],
    "createdAt": 1576874258.137,
    "errors": []
  }
}

```



```
}
```

[Edit halaman ini di GitHub](#)

Menyebarkan node Windows pada kluster EKS

Pelajari cara mengaktifkan dan mengelola dukungan Windows untuk kluster Amazon EKS Anda untuk menjalankan container Windows bersama container Linux.

Pertimbangan

Sebelum men-deploy simpul Windows, perhatikan pertimbangan-pertimbangan berikut.

- Mode Otomatis EKS tidak mendukung node Windows
- Anda dapat menggunakan jaringan host pada node Windows menggunakan HostProcess Pod. Untuk informasi selengkapnya, lihat [Membuat Windows HostProcessPod](#) di dokumentasi Kubernetes.
- Cluster Amazon EKS harus berisi satu atau lebih node Linux atau Fargate untuk menjalankan Pod sistem inti yang hanya berjalan di Linux, seperti CoreDNS.
- Log kube-proxy peristiwa kubelet dan diarahkan ke Log EKS Windows Peristiwa dan diatur ke batas 200 MB.
- Anda tidak dapat menggunakan peran IAM yang sama untuk node Linux dan Windows.
- Anda tidak dapat menggunakan [Tetapkan grup keamanan ke pod individual dengan Pod](#) yang berjalan di node Windows.
- Anda tidak dapat menggunakan [jaringan khusus](#) dengan node Windows.
- Anda tidak dapat menggunakan IPv6 dengan node Windows.
- Simpul Windows mendukung satu antarmuka jaringan elastis per simpul. Secara default, jumlah Pod yang dapat Anda jalankan per node Windows sama dengan jumlah alamat IP yang tersedia per elastic network interface untuk jenis instance node, minus satu. Untuk informasi selengkapnya, lihat [alamat IP per antarmuka jaringan per jenis instans](#) di Panduan EC2 Pengguna Amazon.
- Dalam kluster Amazon EKS, satu layanan dengan load balancer dapat mendukung hingga 1024 Pod back-end. Setiap Pod memiliki alamat IP uniknya sendiri. Batas sebelumnya 64 Pod tidak lagi terjadi, setelah [pembaruan Windows Server](#) dimulai dengan [OS Build 17763.2746](#).
- Kontainer Windows tidak didukung untuk Pod Amazon EKS di Fargate.

- Anda tidak dapat menggunakan Amazon EKS Hybrid Nodes dengan Windows sebagai sistem operasi untuk host.
- Anda tidak dapat mengambil log dari `vpc-resource-controller` Pod. Anda sebelumnya bisa ketika Anda menyebarkan pengontrol ke bidang data.
- Ada periode pendinginan sebelum IPv4 alamat ditetapkan ke Pod baru. Hal ini mencegah lalu lintas mengalir ke Pod lama dengan IPv4 alamat yang sama karena kube-proxy aturan basi.
- Sumber untuk pengontrol dikelola pada GitHub. Untuk berkontribusi, atau mengajukan masalah terhadap pengontrol, kunjungi [proyek](#) di GitHub.
- Saat menentukan ID AMI khusus untuk grup node terkelola Windows, tambahkan `eks:kube-proxy-windows` ke peta konfigurasi AWS IAM Authenticator Anda. Untuk informasi selengkapnya, lihat [the section called “Batas dan ketentuan saat menentukan ID AMI”](#).
- Jika menjaga IPv4 alamat yang tersedia sangat penting untuk subnet Anda, lihat [Panduan Praktik Terbaik EKS - Manajemen Alamat IP Jaringan Windows](#) untuk panduan.
- Pertimbangan untuk Entri Akses EKS
 - Akses Entri untuk digunakan dengan node Windows memerlukan jenis. `EC2_WINDOWS` Untuk informasi selengkapnya, lihat [the section called “Buat entri akses”](#).

Untuk membuat entri akses untuk node Windows:

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn arn:aws:iam::111122223333:role/<role-name> --type EC2_Windows
```

Prasyarat

- Sebuah klaster yang sudah ada.
- Cluster Anda harus memiliki setidaknya satu (kami sarankan setidaknya dua) node Linux atau Fargate Pod untuk menjalankan CoreDNS. Jika Anda mengaktifkan dukungan Windows lama, Anda harus menggunakan node Linux (Anda tidak dapat menggunakan Fargate Pod) untuk menjalankan CoreDNS.
- Peran [IAM cluster Amazon EKS](#) yang ada.

Aktifkan dukungan Windows

1. Jika Anda tidak memiliki node Amazon Linux di klaster dan menggunakan grup keamanan untuk Pod, lanjutkan ke langkah berikutnya. Jika tidak, konfirmasi bahwa kebijakan AmazonEKSVPCResourceController terkelola dilampirkan ke [peran klaster](#) Anda. Ganti *eksClusterRole* dengan nama peran cluster Anda.

```
aws iam list-attached-role-policies --role-name eksClusterRole
```

Contoh output adalah sebagai berikut.

```
{
  "AttachedPolicies": [
    {
      "PolicyName": "AmazonEKSClusterPolicy",
      "PolicyArn": "arn:aws: iam::aws:policy/AmazonEKSClusterPolicy"
    },
    {
      "PolicyName": "AmazonEKSVPCResourceController",
      "PolicyArn": "arn:aws: iam::aws:policy/AmazonEKSVPCResourceController"
    }
  ]
}
```

Jika kebijakan dilampirkan, seperti pada output sebelumnya, lewati langkah berikutnya.

2. Lampirkan kebijakan terkelola [EKSVPCResourcePengontrol Amazon](#) ke [peran IAM klaster Amazon EKS](#) Anda. Ganti *eksClusterRole* dengan nama peran cluster Anda.

```
aws iam attach-role-policy \
  --role-name eksClusterRole \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSVPCResourceController
```

3. Perbarui VPC CNI ConfigMap untuk mengaktifkan Windows IPAM. Catatan, jika VPC CNI diinstal pada klaster Anda menggunakan bagan Helm atau sebagai Add-on Amazon EKS, Anda mungkin tidak dapat langsung memodifikasi. ConfigMap Untuk informasi tentang mengonfigurasi Add-on Amazon EKS, lihat. [the section called “Bidang yang dapat Anda sesuaikan”](#)
 - a. Buat file bernama *vpc-resource-controller-configmap.yaml* dengan isi berikut ini.

```
apiVersion: v1
```

```
kind: ConfigMap
metadata:
  name: amazon-vpc-cni
  namespace: kube-system
data:
  enable-windows-ipam: "true"
```

b. Terapkan ConfigMap ke cluster Anda.

```
kubectl apply -f vpc-resource-controller-configmap.yaml
```

4. Jika cluster Anda memiliki mode otentikasi yang disetel untuk mengaktifkan aws-auth configmap:

- Verifikasi bahwa Anda aws-auth ConfigMap berisi pemetaan untuk peran instance node Windows untuk menyertakan grup izin eks:kube-proxy-windows RBAC. Anda dapat memverifikasi dengan menjalankan perintah berikut.

```
kubectl get configmap aws-auth -n kube-system -o yaml
```

Contoh output adalah sebagai berikut.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: aws-auth
  namespace: kube-system
data:
  mapRoles: |
    - groups:
      - system:bootstrappers
      - system:nodes
      - eks:kube-proxy-windows # This group is required for Windows DNS resolution
        to work
      rolearn: arn:aws: iam::111122223333:role/eksNodeRole
      username: system:node:{{EC2PrivateDNSName}}
  [...]
```

Anda harus melihat eks:kube-proxy-windows terdaftar di bawah grup. Jika grup tidak ditentukan, Anda perlu memperbarui ConfigMap atau membuatnya untuk menyertakan grup yang diperlukan. Untuk informasi lebih lanjut tentang aws-authConfigMap, lihat [the section called “Terapkan aws-authConfigMap ke cluster Anda”](#).

5. Jika cluster Anda memiliki mode otentikasi yang disetel untuk menonaktifkan `aws-auth` configmap, maka Anda dapat menggunakan Entri Akses EKS. Buat peran node baru untuk digunakan dengan instance Windows, dan EKS akan secara otomatis membuat entri akses tipe `EC2_WINDOWS`.

Menyebarkan Pod Windows

Saat Anda menerapkan Pod ke klaster Anda, Anda perlu menentukan sistem operasi yang mereka gunakan jika Anda menjalankan campuran tipe node.

Untuk Pod Linux, gunakan teks pemilih node berikut dalam manifes Anda.

```
nodeSelector:
  kubernetes.io/os: linux
  kubernetes.io/arch: amd64
```

Untuk Pod Windows, gunakan teks pemilih node berikut dalam manifes Anda.

```
nodeSelector:
  kubernetes.io/os: windows
  kubernetes.io/arch: amd64
```

Anda dapat menerapkan [aplikasi sampel](#) untuk melihat pemilih node yang digunakan.

Mendukung kepadatan Pod yang lebih tinggi pada node Windows

Di Amazon EKS, setiap Pod dialokasikan IPv4 alamat dari VPC Anda. Karena hal ini, jumlah Pod yang dapat Anda gunakan ke sebuah node dibatasi oleh alamat IP yang tersedia, bahkan jika ada sumber daya yang cukup untuk menjalankan lebih banyak Pod pada node. Karena hanya satu elastic network interface yang didukung oleh node Windows, secara default, jumlah maksimum alamat IP yang tersedia pada node Windows sama dengan:

```
Number of private IPv4 addresses for each interface on the node - 1
```

Satu alamat IP digunakan sebagai alamat IP utama dari antarmuka jaringan, sehingga tidak dapat dialokasikan ke Pod.

Anda dapat mengaktifkan kepadatan Pod yang lebih tinggi pada node Windows dengan mengaktifkan delegasi awalan IP. Fitur ini memungkinkan Anda untuk menetapkan /28 IPv4 awalan

ke antarmuka jaringan utama, bukan menetapkan alamat sekunder. IPv4 Menetapkan awalan IP meningkatkan IPv4 alamat maksimum yang tersedia pada node menjadi:

```
(Number of private IPv4 addresses assigned to the interface attached to the node - 1) * 16
```

Dengan jumlah alamat IP yang tersedia secara signifikan lebih besar ini, alamat IP yang tersedia seharusnya tidak membatasi kemampuan Anda untuk menskalakan jumlah Pod pada node Anda. Lihat informasi yang lebih lengkap di [the section called “Meningkatkan alamat IP”](#).

Nonaktifkan dukungan Windows

1. Jika kluster Anda berisi node Amazon Linux dan Anda menggunakan [grup keamanan untuk Pod](#) bersamanya, lewati langkah ini.

Hapus kebijakan IAM AmazonVPCResourceController terkelola dari [peran kluster](#) Anda. Ganti *eksClusterRole* dengan nama peran cluster Anda.

```
aws iam detach-role-policy \
  --role-name eksClusterRole \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSVPCResourceController
```

2. Nonaktifkan Windows IPAM di. amazon-vpc-cni ConfigMap

```
kubectl patch configmap/amazon-vpc-cni \
  -n kube-system \
  --type merge \
  -p '{"data":{"enable-windows-ipam":"false"}}'
```

Menyebarkan kluster pribadi dengan akses internet terbatas

Topik ini menjelaskan cara menerapkan kluster Amazon EKS yang diterapkan di AWS Cloud, tetapi tidak memiliki akses internet keluar. Jika Anda memiliki cluster lokal di AWS Outposts, lihat [the section called “Simpulan”](#), bukan topik ini.

Jika Anda tidak terbiasa dengan jaringan Amazon EKS, lihat [De-mystifying jaringan cluster untuk node pekerja Amazon EKS](#). Jika kluster Anda tidak memiliki akses internet keluar, maka kluster harus memenuhi persyaratan berikut:

- Cluster Anda harus menarik gambar dari registri kontainer yang ada di VPC Anda. Anda dapat membuat Amazon Elastic Container Registry di VPC Anda dan menyalin gambar kontainer ke sana untuk diambil node Anda. Untuk informasi selengkapnya, lihat [the section called “Salin gambar ke repositori”](#).
- Cluster Anda harus mengaktifkan akses pribadi endpoint. Hal ini diperlukan untuk node untuk mendaftar dengan endpoint cluster. titik akhir akses publik adalah opsional. Untuk informasi selengkapnya, lihat [the section called “Akses titik akhir klaster”](#).
- Node Linux dan Windows yang dikelola sendiri harus menyertakan argumen bootstrap berikut sebelum diluncurkan. Argumen ini melewati introspeksi Amazon EKS dan tidak memerlukan akses ke Amazon EKS API dari dalam VPC.
 - a. Tentukan nilai endpoint cluster Anda dengan perintah berikut. Ganti *my-cluster* dengan nama klaster Anda.

```
aws eks describe-cluster --name my-cluster --query cluster.endpoint --output text
```

Contoh output adalah sebagai berikut.

```
https://EXAMPLE108C897D9B2F1B21D5EXAMPLE.sk1.region-code.eks.amazonaws.com
```

- b. Tentukan nilai otoritas sertifikat klaster Anda dengan perintah berikut. Ganti *my-cluster* dengan nama klaster Anda.

```
aws eks describe-cluster --name my-cluster --query cluster.certificateAuthority --output text
```

Output yang dikembalikan adalah string panjang.

- c. Ganti *cluster-endpoint* dan *certificate-authority* dalam perintah berikut dengan nilai yang dikembalikan dalam output dari perintah sebelumnya. Untuk informasi selengkapnya tentang menentukan argumen bootstrap saat meluncurkan node yang dikelola sendiri, lihat [the section called “Amazon Linux”](#) dan [the section called “Windows”](#)

- Untuk node Linux:

```
--apiserver-endpoint cluster-endpoint --b64-cluster-ca certificate-authority
```

Untuk argumen tambahan, lihat [skrip bootstrap](#) pada GitHub.

- Untuk node Windows:

Note

Jika Anda menggunakan layanan khusus CIDR, maka Anda perlu menentukannya menggunakan `-ServiceCIDR` parameter. Jika tidak, resolusi DNS untuk Pod di klaster akan gagal.

```
-APIServerEndpoint cluster-endpoint -Base64ClusterCA certificate-authority
```

Untuk argumen tambahan, lihat [the section called “Parameter konfigurasi skrip bootstrap”](#).

- Cluster Anda `aws-auth ConfigMap` harus dibuat dari dalam VPC Anda. Untuk informasi selengkapnya tentang membuat dan menambahkan entri ke `aws-auth ConfigMap`, masukkan `eksctl create iamidentitymapping --help` di terminal Anda. Jika `ConfigMap` tidak ada di server Anda, `eksctl` akan membuatnya ketika Anda menggunakan perintah untuk menambahkan pemetaan identitas.
- Pod yang dikonfigurasi dengan [peran IAM untuk akun layanan](#) memperoleh kredensial dari panggilan API AWS Security Token Service (AWS STS). Jika tidak ada akses internet keluar, Anda harus membuat dan menggunakan titik akhir VPC AWS STS di VPC Anda. Sebagian besar AWS v1 SDKs menggunakan endpoint AWS STS global secara default (`sts.amazonaws.com`), yang tidak menggunakan titik akhir AWS STS VPC. Untuk menggunakan titik akhir VPC AWS STS, Anda mungkin perlu mengonfigurasi SDK Anda untuk menggunakan endpoint AWS STS regional (`sts.region-code.amazonaws.com`). Untuk informasi selengkapnya, lihat [the section called “Titik akhir STS”](#).
- Subnet VPC klaster Anda harus memiliki titik akhir antarmuka VPC untuk AWS layanan apa pun yang perlu diakses oleh Pod Anda. Untuk informasi selengkapnya, lihat [Mengakses AWS layanan menggunakan titik akhir VPC antarmuka](#). Beberapa layanan dan titik akhir yang umum digunakan tercantum dalam tabel berikut. Untuk daftar lengkap titik akhir, lihat [AWS layanan yang terintegrasi dengan AWS PrivateLink](#) dalam [AWS PrivateLink Panduan](#).

Kami menyarankan Anda [mengaktifkan nama DNS pribadi](#) untuk titik akhir VPC Anda, sehingga beban kerja dapat terus menggunakan titik akhir layanan AWS publik tanpa masalah.

Layanan	Titik Akhir
Amazon EC2	<code>com.amazonaws.<i>region-code</i>.ec2</code>

Layanan	Titik Akhir
Amazon Elastic Container Registry (untuk menarik gambar kontainer)	com.amazonaws. <i>region-code</i> .ecr.api, com.amazonaws. <i>region-code</i> .ecr.dkr, dan com.amazonaws. <i>region-code</i> .s3
Aplikasi Load Balancer dan Network Load Balancer	com.amazonaws. <i>region-code</i> .elasticloadbalancing
AWS X-Ray	com.amazonaws. <i>region-code</i> .xray
CloudWatch Log Amazon	com.amazonaws. <i>region-code</i> .log
AWS Security Token Service (diperlukan saat menggunakan peran IAM untuk akun layanan)	com.amazonaws. <i>region-code</i> .sts
Amazon EKS Auth (diperlukan saat menggunakan asosiasi Pod Identity)	com.amazonaws. <i>region-code</i> .eks-auth
Amazon EKS	com.amazonaws. <i>region-code</i> .eks

- Setiap node yang dikelola sendiri harus disebarkan ke subnet yang memiliki titik akhir antarmuka VPC yang Anda butuhkan. Jika Anda membuat grup node terkelola, grup keamanan titik akhir antarmuka VPC harus mengizinkan CIDR untuk subnet, atau Anda harus menambahkan grup keamanan node yang dibuat ke grup keamanan titik akhir antarmuka VPC.
- Jika Pod Anda menggunakan volume Amazon EFS, maka sebelum menerapkan [sistem file elastis Store dengan Amazon EFS](#), file [kustomization.yaml](#) driver harus diubah untuk mengatur image container agar menggunakan Region AWS yang sama dengan cluster Amazon EKS.
- Route53 tidak mendukung. AWS PrivateLink Anda tidak dapat mengelola catatan DNS Route53 dari kluster Amazon EKS pribadi. [Ini berdampak pada Kubernetes external-dns.](#)
- Jika Anda menggunakan AMI EKS Optimized, Anda harus mengaktifkan ec2 endpoint pada tabel di atas. Atau, Anda dapat secara manual mengatur nama DNS Node. AMI yang dioptimalkan digunakan EC2 APIs untuk mengatur nama DNS node secara otomatis.
- Anda dapat menggunakan [AWS Load Balancer Controller](#) untuk menyebarkan AWS Application Load Balancers (ALB) dan Network Load Balancer ke cluster pribadi Anda. Saat menerapkannya, Anda harus menggunakan [flag baris perintah](#) untuk mengaturnable-shield,enable-waf, dan enable-wafv2 ke false. [Penemuan sertifikat](#) dengan nama host dari objek Ingress tidak

didukung. Ini karena pengontrol perlu mencapai AWS Certificate Manager, yang tidak memiliki titik akhir antarmuka VPC.

Pengendali yang didukung pada penyeimbang beban jaringan dengan target IP, yang diperlukan untuk penggunaan pada Fargate. Untuk informasi selengkapnya, silakan lihat [the section called “Penyeimbangan beban aplikasi”](#) dan [the section called “Buat penyeimbang beban jaringan”](#).

- [Cluster Autoscaler didukung](#). Saat menerapkan Cluster Autoscaler Pod, pastikan bahwa baris perintah termasuk `--aws-use-static-instance-list=true` Untuk informasi selengkapnya, lihat [Menggunakan Daftar Instans Statis](#) pada GitHub. VPC node pekerja juga harus menyertakan titik akhir VPC STS dan titik akhir AWS VPC penskalaan otomatis.
- Beberapa produk perangkat lunak kontainer menggunakan panggilan API yang mengakses AWS Marketplace Metering Service untuk memantau penggunaan. Cluster pribadi tidak mengizinkan panggilan ini, jadi Anda tidak dapat menggunakan jenis penampung ini di kluster pribadi.

Memahami siklus hidup versi Kubernetes di EKS

Kubernetes berkembang pesat dengan fitur-fitur baru, pembaruan desain, dan perbaikan bug. Komunitas merilis versi minor Kubernetes baru (seperti 1.33) rata-rata setiap empat bulan sekali. Amazon EKS mengikuti siklus rilis dan penghentian upstream untuk versi minor. Saat versi Kubernetes baru tersedia di Amazon EKS, kami sarankan supaya Anda memperbarui kluster secara proaktif untuk menggunakan versi terbaru yang tersedia.

Versi minor berada di bawah dukungan standar di Amazon EKS selama 14 bulan pertama setelah dirilis. Setelah versi melewati akhir tanggal dukungan standar, ia memasuki dukungan diperpanjang untuk 12 bulan ke depan. Dukungan yang diperluas memungkinkan Anda untuk tinggal di versi Kubernetes tertentu lebih lama dengan biaya tambahan per jam cluster. Jika Anda belum memperbarui kluster sebelum periode dukungan yang diperpanjang berakhir, kluster Anda akan ditingkatkan secara otomatis ke versi perpanjangan tertua yang saat ini didukung.

Dukungan diperpanjang diaktifkan secara default. [the section called “Nonaktifkan dukungan yang diperluas”](#)

Kami menyarankan Anda membuat kluster Anda dengan versi Kubernetes terbaru yang tersedia yang didukung oleh Amazon EKS. Jika aplikasi Anda memerlukan versi Kubernetes tertentu, Anda dapat memilih versi yang lebih lama. Anda dapat membuat kluster Amazon EKS baru pada versi apa pun yang ditawarkan dalam dukungan standar atau diperpanjang.

Versi yang tersedia pada dukungan standar

Versi Kubernetes berikut saat ini tersedia di dukungan standar Amazon EKS:

- 1.33
- 1.32
- 1.31
- 1.30

Untuk perubahan penting yang harus diperhatikan untuk setiap versi dalam dukungan standar, lihat [the section called “Versi dukungan standar”](#).

Versi yang tersedia pada dukungan diperpanjang

Versi Kubernetes berikut saat ini tersedia di dukungan diperpanjang Amazon EKS:

- 1.29
- 1.28
- 1.27

Untuk perubahan penting yang harus diperhatikan untuk setiap versi dalam dukungan yang diperluas, lihat [the section called “Versi dukungan yang diperluas”](#).

Kalender rilis Amazon EKS Kubernetes

Tabel berikut menunjukkan tanggal rilis dan dukungan penting untuk dipertimbangkan untuk setiap versi Kubernetes. Penagihan untuk dukungan diperpanjang dimulai pada awal hari ketika versi mencapai akhir dukungan standar.

Note

Tanggal yang berupa hanya bulan dan tahun merupakan perkiraan, dan akan diperbarui dengan tanggal persisnya saat diketahui.

Untuk menerima pemberitahuan semua perubahan file sumber ke halaman dokumentasi khusus ini, Anda dapat berlangganan URL berikut dengan pembaca RSS:

<https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/clusters/kubernetes-versions.adoc.atom>

Versi Kubernetes	Rilis hulu	Rilis Amazon EKS	Akhir dari dukungan standar	Akhir dari dukungan yang diperpanjang
1.33	April 23, 2025	29 Mei 2025	Juli 29, 2026	Juli 29, 2027
1.32	Desember 11, 2024	Januari 23, 2025	Maret 23, 2026	Maret 23, 2027
1.31	Agustus 13, 2024	September 26, 2024	November 26, 2025	November 26, 2026
1.30	April 17, 2024	23 Mei 2024	Juli 23, 2025	Juli 23, 2026
1.29	13 Desember 2023	23 Januari 2024	Maret 23, 2025	Maret 23, 2026
1.28	15 Agustus 2023	26 September 2023	November 26, 2024	November 26, 2025
1.27	11 April 2023	24 Mei 2023	Juli 24, 2024	Juli 24, 2025

Dapatkan informasi versi dengan AWS CLI

Anda dapat menggunakan AWS CLI untuk mendapatkan informasi tentang versi Kubernetes yang tersedia di EKS, seperti tanggal akhir Standard Support.

Untuk mengambil informasi tentang versi Kubernetes yang tersedia di EKS menggunakan CLI AWS

1. Buka terminal Anda.
2. Pastikan AWS CLI Anda telah diinstal dan dikonfigurasi. Untuk informasi selengkapnya, lihat [Menginstal atau memperbarui ke versi terbaru CLI](#).
3. Jalankan perintah berikut:

```
aws eks describe-cluster-versions
```

4. Perintah akan mengembalikan output JSON dengan rincian tentang versi cluster yang tersedia. Berikut adalah contoh output:

```
{
  "clusterVersions": [
    {
      "clusterVersion": "1.31",
      "clusterType": "eks",
      "defaultPlatformVersion": "eks.21",
      "defaultVersion": true,
      "releaseDate": "2024-09-25T17:00:00-07:00",
      "endOfStandardSupportDate": "2025-11-25T16:00:00-08:00",
      "endOfExtendedSupportDate": "2026-11-25T16:00:00-08:00",
      "status": "STANDARD_SUPPORT",
      "kubernetesPatchVersion": "1.31.3"
    }
  ]
}
```

Output memberikan informasi berikut untuk setiap versi cluster:

- `clusterVersion`: Versi Kubernetes dari cluster EKS
- `clusterType`: Jenis cluster (misalnya, "eks")
- `defaultPlatformVersion`: Versi platform EKS default
- `defaultVersion`: Apakah ini versi default
- `releaseDate`: Tanggal ketika versi ini dirilis
- `endOfStandardSupportDate`: Tanggal ketika dukungan standar berakhir
- `endOfExtendedSupportDate`: Tanggal ketika dukungan diperpanjang berakhir
- `status`: Status dukungan versi saat ini, seperti `STANDARD_SUPPORT` atau `EXTENDED_SUPPORT`
- `kubernetesPatchVersion`: Versi patch Kubernetes tertentu

Versi Amazon EKS FAQs

Berapa banyak versi Kubernetes yang tersedia dalam dukungan standar?

Sejalan dengan dukungan komunitas Kubernetes untuk versi Kubernetes, Amazon EKS berkomitmen untuk menawarkan dukungan untuk tiga versi Kubernetes pada waktu tertentu. Kami akan mengumumkan akhir dari tanggal dukungan standar dari versi minor Kubernetes yang diberikan setidaknya 60 hari sebelumnya. Karena kualifikasi Amazon EKS dan proses rilis untuk versi Kubernetes baru, akhir tanggal dukungan standar versi Kubernetes di Amazon EKS akan berakhir setelah tanggal proyek Kubernetes berhenti mendukung versi upstream.

Berapa lama Kubernetes menerima dukungan standar dari Amazon EKS?

Versi Kubernetes menerima dukungan standar selama 14 bulan setelah pertama kali tersedia di Amazon EKS. Ini benar bahkan jika Kubernetes upstream tidak lagi mendukung versi yang tersedia di Amazon EKS. Kami melakukan backport patch keamanan yang berlaku untuk versi Kubernetes yang didukung di Amazon EKS.

Apakah saya diberi tahu ketika dukungan standar berakhir untuk versi Kubernetes di Amazon EKS?

Ya. Jika ada cluster di akun Anda yang menjalankan versi mendekati akhir dukungan, Amazon EKS mengirimkan pemberitahuan melalui Dasbor AWS Kesehatan sekitar 12 bulan setelah versi Kubernetes dirilis di Amazon EKS. Pemberitahuan tersebut mencakup akhir tanggal dukungan. Ini setidaknya 60 hari sejak tanggal pemberitahuan.

Fitur Kubernetes mana yang didukung oleh Amazon EKS?

Amazon EKS mendukung semua fitur yang tersedia secara umum (GA) dari Kubernetes API. Beta baru APIs tidak diaktifkan di cluster secara default. Namun, versi beta yang ada sebelumnya APIs dan versi baru dari beta yang ada APIs terus diaktifkan secara default. Fitur alfa tidak didukung.

Apakah grup node terkelola Amazon EKS secara otomatis diperbarui bersama dengan versi bidang kontrol cluster?

Tidak. Grup node terkelola membuat EC2 instance Amazon di akun Anda. Instans ini tidak ditingkatkan secara otomatis saat Anda atau Amazon EKS memperbarui pesawat kontrol Anda. Untuk informasi selengkapnya, lihat [the section called “Perbarui”](#). Sebaiknya pertahankan versi Kubernetes yang sama di bidang kendali dan simpul Anda.

Apakah grup node yang dikelola sendiri secara otomatis diperbarui bersama dengan versi bidang kontrol cluster?

Tidak. Grup node yang dikelola sendiri menyertakan EC2 instans Amazon di akun Anda. Instans ini tidak ditingkatkan secara otomatis saat Anda atau Amazon EKS memperbarui versi pesawat kontrol atas nama Anda. Grup node yang dikelola sendiri tidak memiliki indikasi apa pun di konsol bahwa ia perlu diperbarui. Anda dapat melihat versi kubelet yang diinstal pada sebuah simpul dengan memilih simpul di dalam daftar Simpul pada tab Gambaran Umum klaster Anda untuk menentukan simpul-simpul yang perlu diperbarui. Anda harus memperbarui simpul secara manual. Untuk informasi selengkapnya, lihat [the section called “Metode pembaruan”](#).

Proyek Kubernetes menguji kompatibilitas antara bidang kontrol dan node hingga tiga versi minor. Misalnya, 1.30 node terus beroperasi ketika diatur oleh bidang kontrol 1.33. Namun, menjalankan cluster dengan node yang terus-menerus tiga versi minor di belakang bidang kontrol tidak disarankan. Untuk informasi selengkapnya, lihat [kebijakan dukungan versi Kubernetes dan versi miring](#) dalam dokumentasi Kubernetes. Sebaiknya pertahankan versi Kubernetes yang sama di bidang kendali dan simpul Anda.

Apakah Pod yang berjalan di Fargate secara otomatis ditingkatkan dengan upgrade versi bidang kontrol cluster otomatis?

Tidak. Kami sangat menyarankan menjalankan Fargate Pods sebagai bagian dari pengontrol replikasi, seperti penerapan Kubernetes. Kemudian lakukan restart bergulir dari semua Pod Fargate. Versi baru dari Fargate Pod di-deploy dengan kubelet versi yang sama dengan versi cluster control plane Anda yang diperbarui. Untuk informasi selengkapnya, lihat [Deployment](#) dalam dokumentasi Kubernetes.

 Important

Jika Anda memperbarui bidang kontrol, Anda masih harus memperbarui node Fargate sendiri. Untuk memperbarui node Fargate, hapus Pod Fargate yang diwakili oleh node dan redeploy Pod. Pod baru di-deploy dengan kubelet versi yang sama dengan klaster Anda.

Versi Kubernetes apa yang didukung untuk node hybrid?

Amazon EKS Hybrid Nodes mendukung versi Kubernetes yang sama dengan cluster Amazon EKS dengan tipe komputasi node lainnya, termasuk dukungan versi Kubernetes standar dan

diperpanjang. Node hibrida tidak ditingkatkan secara otomatis saat Anda memutakhirkan versi pesawat kontrol Anda dan Anda bertanggung jawab untuk meningkatkan node hibrida Anda. Untuk informasi selengkapnya, lihat [the section called “Tingkatkan node hibrida”](#).

Amazon EKS memperpanjang dukungan FAQs

Dukungan standar dan terminologi dukungan yang diperluas adalah hal baru bagi saya. Apa arti istilah-istilah itu?

Dukungan standar untuk versi Kubernetes di Amazon EKS dimulai ketika versi Kubernetes dirilis di Amazon EKS, dan akan berakhir 14 bulan setelah tanggal rilis. Dukungan tambahan untuk versi Kubernetes akan dimulai segera setelah akhir dukungan standar, dan akan berakhir setelah 12 bulan ke depan. Misalnya, dukungan standar untuk versi 1.23 di Amazon EKS berakhir pada 11 Oktober 2023. Dukungan diperpanjang untuk versi 1.23 dimulai pada 12 Oktober 2023 dan berakhir pada 11 Oktober 2024.

Apa yang harus saya lakukan untuk mendapatkan dukungan tambahan untuk cluster Amazon EKS?

Anda harus mengaktifkan dukungan yang diperluas (lihat [the section called “Aktifkan dukungan yang diperluas”](#)) untuk kluster Anda dengan mengubah kebijakan pemutakhiran kluster menjadi EXTENDED. Secara default, untuk semua kluster baru dan yang sudah ada, kebijakan pemutakhiran disetel ke EXTENDED, kecuali ditentukan lain. Lihat [the section called “Lihat kebijakan peningkatan”](#) untuk melihat kebijakan pemutakhiran untuk kluster Anda. Dukungan standar akan dimulai ketika versi Kubernetes dirilis di Amazon EKS, dan akan berakhir 14 bulan setelah tanggal rilis. Dukungan tambahan untuk versi Kubernetes akan dimulai segera setelah akhir dukungan standar, dan akan berakhir setelah 12 bulan ke depan.

Untuk versi Kubernetes mana saya bisa mendapatkan dukungan tambahan?

Anda dapat menjalankan cluster pada versi apa pun hingga 12 bulan setelah akhir dukungan standar untuk versi itu. Ini berarti bahwa setiap versi akan didukung selama 26 bulan di Amazon EKS (14 bulan dukungan standar ditambah 12 bulan dukungan diperpanjang).

Bagaimana jika saya tidak ingin menggunakan dukungan tambahan?

Jika Anda tidak ingin secara otomatis terdaftar dalam dukungan yang diperluas, Anda dapat meningkatkan kluster Anda ke versi Kubernetes yang ada dalam dukungan Amazon EKS standar. Lihat [the section called “Nonaktifkan dukungan yang diperluas”](#) untuk mempelajari cara menonaktifkan dukungan yang diperluas. Catatan: Jika Anda menonaktifkan dukungan yang diperluas, kluster Anda akan ditingkatkan secara otomatis di akhir dukungan standar.

Apa yang akan terjadi pada akhir 12 bulan dukungan diperpanjang?

Cluster yang berjalan pada versi Kubernetes yang telah menyelesaikan siklus hidup 26 bulan (14 bulan dukungan standar ditambah 12 bulan dukungan diperpanjang) akan otomatis ditingkatkan ke versi berikutnya. Upgrade otomatis hanya mencakup bidang kontrol Kubernetes. Jika Anda memiliki node Mode Otomatis EKS, mereka dapat memperbarui secara otomatis. Node yang dikelola sendiri dan Grup Node Terkelola EKS akan tetap ada di versi sebelumnya.

Di akhir tanggal dukungan yang diperpanjang, Anda tidak dapat lagi membuat kluster Amazon EKS baru dengan versi yang tidak didukung. Pesawat kontrol yang ada diperbarui secara otomatis oleh Amazon EKS ke versi paling awal yang didukung melalui proses penerapan bertahap setelah tanggal dukungan berakhir. Setelah pembaruan bidang kontrol otomatis, pastikan untuk memperbarui add-on cluster dan EC2 node Amazon secara manual. Untuk informasi selengkapnya, lihat [the section called “Perbarui versi Kubernetes”](#).

Kapan tepatnya pesawat kontrol saya diperbarui secara otomatis setelah akhir tanggal dukungan yang diperpanjang?

Amazon EKS tidak dapat memberikan kerangka waktu tertentu. Pembaruan otomatis dapat terjadi kapan saja setelah akhir tanggal dukungan yang diperpanjang. Anda tidak akan menerima pemberitahuan apa pun sebelum pembaruan. Kami menyarankan Anda secara proaktif memperbarui pesawat kontrol Anda tanpa bergantung pada proses pembaruan otomatis Amazon EKS. Untuk informasi selengkapnya, lihat [the section called “Perbarui versi Kubernetes”](#).

Bisakah saya meninggalkan pesawat kontrol saya pada versi Kubernetes tanpa batas waktu?

Tidak. Keamanan cloud di AWS adalah prioritas tertinggi. Melewati titik tertentu (biasanya satu tahun), komunitas Kubernetes berhenti merilis patch common vulnerabilities and exposure (CVE) dan mencegah pengiriman CVE untuk versi yang tidak didukung. Ini berarti bahwa kerentanan khusus untuk versi Kubernetes yang lebih lama bahkan mungkin tidak dilaporkan. Ini membuat cluster terbuka tanpa pemberitahuan dan tidak ada opsi perbaikan jika terjadi kerentanan. Mengingat hal ini, Amazon EKS tidak mengizinkan pesawat kontrol untuk tetap menggunakan versi yang mencapai akhir dukungan yang diperpanjang.

Apakah ada biaya tambahan untuk mendapatkan dukungan tambahan?

Ya, ada biaya tambahan untuk kluster Amazon EKS yang berjalan dalam dukungan yang diperpanjang. [Untuk detail harga, lihat dukungan tambahan Amazon EKS untuk harga versi Kubernetes di AWS blog atau halaman harga kami.](#)

Apa yang termasuk dalam dukungan diperpanjang?

Cluster Amazon EKS di Extended Support menerima patch keamanan berkelanjutan untuk pesawat kontrol Kubernetes. Selain itu, Amazon EKS akan merilis tambalan untuk Amazon VPC CNikube-proxy,, dan add-on CoreDNS untuk versi Extended Support. Amazon EKS juga akan merilis tambalan untuk AWS Amazon EKS yang diterbitkan yang dioptimalkan untuk AMIs Amazon Linux, Bottlerocket, dan Windows, serta node Amazon EKS Fargate untuk versi tersebut. Semua cluster di Extended Support akan terus mendapatkan akses ke dukungan teknis dari AWS.

Apakah ada batasan untuk patch untuk komponen non-Kubernetes dalam dukungan yang diperluas?

Sementara Extended Support mencakup semua komponen khusus Kubernetes AWS, itu hanya akan memberikan dukungan untuk Amazon EKS yang dipublikasikan yang dioptimalkan untuk AWS AMIs Amazon Linux, Bottlerocket, dan Windows setiap saat. Ini berarti, Anda berpotensi memiliki komponen yang lebih baru (seperti OS atau kernel) di AMI yang dioptimalkan Amazon EKS saat menggunakan Extended Support. Misalnya, setelah Amazon Linux 2 mencapai [akhir siklus hidupnya pada tahun 2025](#), Amazon EKS yang dioptimalkan Amazon Linux AMIs akan dibangun menggunakan OS Linux Amazon yang lebih baru. Amazon EKS akan mengumumkan dan mendokumentasikan perbedaan siklus hidup dukungan penting seperti ini untuk setiap versi Kubernetes.

Dapatkah saya membuat cluster baru menggunakan versi pada dukungan yang diperluas?

Ya.

Tinjau catatan rilis untuk versi Kubernetes pada dukungan standar

Topik ini memberikan perubahan penting yang harus diperhatikan untuk setiap versi Kubernetes dalam dukungan standar. Saat memutakhirkan, tinjau dengan cermat perubahan yang terjadi antara versi lama dan baru untuk klaster Anda.

Kubernetes 1.33

Kubernetes sekarang 1.33 tersedia di Amazon EKS. Untuk informasi lebih lanjut tentang Kubernetes 1.33, lihat pengumuman rilis [resmi](#).

Important

- Dynamic Resource Allocation beta Kubernetes API diaktifkan.

- API beta ini meningkatkan pengalaman penjadwalan dan pemantauan beban kerja yang membutuhkan sumber daya seperti GPUs
- API beta didefinisikan oleh komunitas Kubernetes, dan mungkin berubah di Kubernetes versi masa depan.
- Tinjau secara cermat [tahapan Fitur](#) dalam dokumentasi Kubernetes untuk memahami implikasi penggunaan beta APIs
- AWS tidak merilis Amazon Linux 2 AMI yang dioptimalkan EKS untuk Kubernetes 1.33.
- AWS mendorong Anda untuk bermigrasi ke Amazon Linux 2023. Pelajari cara [the section called “Tingkatkan ke AL2 023”](#).
- Untuk informasi selengkapnya, lihat [the section called “Penghentian Amazon Linux 2 AMI”](#).

- In-Place Pod Resource Resize (Beta): Pengubahan ukuran sumber daya di tempat telah dipromosikan menjadi beta, memungkinkan pembaruan dinamis ke CPU dan sumber daya memori untuk Pod yang ada tanpa restart - memungkinkan penskalaan vertikal beban kerja stateful tanpa waktu henti dan penyesuaian sumber daya yang mulus berdasarkan pola lalu lintas.
- Kontainer Sidecar Sekarang Stabil: Kontainer sidecar telah lulus menjadi stabil, mengimplementasikan sidecar sebagai wadah init khusus dengan `restartPolicy: Always` start sebelum wadah aplikasi, berjalan di seluruh siklus hidup pod, dan mendukung probe untuk pensinyalan status operasional.
 - Untuk informasi selengkapnya, lihat [Kontainer Sidecar](#) di Dokumentasi Kubernetes.
- Endpoints API Deprecation: Endpoints API sekarang secara resmi tidak digunakan lagi dan akan mengembalikan peringatan saat diakses - memigrasikan beban kerja dan skrip untuk menggunakan EndpointSlices API sebagai gantinya, yang mendukung fitur modern seperti jaringan dual-stack dan menangani beberapa per Layanan. EndpointSlices
 - Untuk informasi selengkapnya, lihat [Kubernetes v1.33: Melanjutkan transisi dari Endpoint ke EndpointSlice](#) Blog Kubernetes.
- Dukungan Adaptor Kain Elastis: Grup keamanan default untuk kluster Amazon EKS sekarang mendukung lalu lintas Elastic Fabric Adapter (EFA). Grup keamanan default memiliki aturan keluar baru yang memungkinkan lalu lintas EFA dengan tujuan grup keamanan yang sama. Hal ini memungkinkan lalu lintas EFA dalam cluster.
 - Untuk informasi selengkapnya, lihat [Adaptor Kain Elastis untuk beban kerja AI/ML dan HPC di Amazon EC2 di Panduan Pengguna Amazon](#) Elastic Compute Cloud.

Untuk 1.33 changelog Kubernetes yang lengkap, lihat -1.33.md <https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG>

Kubernetes 1.32

Kubernetes sekarang 1.32 tersedia di Amazon EKS. Untuk informasi lebih lanjut tentang Kubernetes 1.32, lihat pengumuman rilis [resmi](#).

Important

- Versi `flowcontrol.apiserver.k8s.io/v1beta3` API FlowSchema dan `PriorityLevelConfiguration` telah dihapus dalam versi 1.32. Jika Anda menggunakan ini APIs, Anda harus memperbarui konfigurasi Anda untuk menggunakan versi terbaru yang didukung sebelum memutakhirkan.
 - `ServiceAccount metadata.annotations[kubernetes.io/enforce-mountable-secrets]` telah usang dalam versi 1.32 dan akan dihapus dalam rilis versi minor Kubernetes mendatang. Disarankan untuk menggunakan ruang nama terpisah untuk mengisolasi akses ke rahasia yang dipasang.
 - Versi Kubernetes 1.32 adalah versi terakhir di mana Amazon EKS akan merilis Amazon Linux 2 (AL2) AMIs. Dari versi 1.33 seterusnya, Amazon EKS akan terus merilis Amazon Linux 2023 (AL2023) dan berbasis Bottlerocket AMIs.
-
- Fitur Memory Manager telah lulus ke status Generally Available (GA) dalam versi Kubernetes 1.32. Peningkatan ini memberikan alokasi memori yang lebih efisien dan dapat diprediksi untuk aplikasi kontainer, terutama bermanfaat untuk beban kerja dengan persyaratan memori tertentu.
 - `PersistentVolumeClaims (PVCs)` dibuat `StatefulSets` sekarang termasuk fungsionalitas pembersihan otomatis. Ketika PVCs tidak lagi diperlukan, mereka akan dihapus secara otomatis sambil mempertahankan persistensi data selama `StatefulSet` pembaruan dan operasi pemeliharaan node. Fitur ini menyederhanakan manajemen penyimpanan dan membantu mencegah yatim piatu di cluster Anda. PVCs
 - Fungsionalitas Pemilih Bidang Sumber Daya Kustom telah diperkenalkan, memungkinkan pengembang untuk menambahkan pemilih bidang ke sumber daya khusus. Fitur ini menyediakan kemampuan penyaringan yang sama yang tersedia untuk objek Kubernetes bawaan ke sumber daya kustom, memungkinkan pemfilteran sumber daya yang lebih tepat dan efisien serta mempromosikan praktik desain API yang lebih baik.

Untuk 1.32 changelog Kubernetes yang lengkap, lihat [-1.32.md https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG](https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG)

Perubahan otentikasi anonim

Dimulai dengan Amazon EKS 1.32, otentikasi anonim dibatasi pada titik akhir pemeriksaan kesehatan server API berikut:

- `/healthz`
- `/livez`
- `/readyz`

Permintaan ke titik akhir lain yang menggunakan `system:unauthenticated` pengguna akan menerima respons `401 Unauthorized HTTP`. Peningkatan keamanan ini membantu mencegah akses kluster yang tidak diinginkan yang dapat terjadi karena kebijakan RBAC yang salah dikonfigurasi.

Note

Peran `public-info-viewer` RBAC terus berlaku untuk titik akhir pemeriksaan kesehatan yang tercantum di atas.

Penghentian Amazon Linux 2 AMI

Untuk Kubernetes versi 1.33 dan yang lebih baru, EKS tidak akan menyediakan Amazon Machine Images (AMI) Amazon Linux 2 (AL2) yang telah dioptimalkan sebelumnya (AL2). AMIs

AWS menyarankan mengadopsi Mode Otomatis EKS, atau bermigrasi ke sistem operasi yang lebih baru, seperti Amazon Linux 2023 (AL2023) atau Bottlerocket.

- [the section called “Migrasi dari MNGs”](#)
- [the section called “Tingkatkan ke AL2 023”](#)
- [the section called “Bottlerocket”](#)

 Note

Pembaruan ini berlaku untuk yang dioptimalkan EKS AL2 AMIs. Untuk informasi lebih lanjut tentang sistem operasi itu sendiri, lihat [Amazon Linux 2 FAQs](#).

Kubernetes 1.31

Kubernetes sekarang 1.31 tersedia di Amazon EKS. Untuk informasi lebih lanjut tentang Kubernetes 1.31, lihat pengumuman rilis [resmi](#).

 Important

- `--keep-terminated-pod-volumesFlag` kubelet yang tidak digunakan lagi sejak 2017 telah dihapus sebagai bagian dari rilis versi 1.31. Perubahan ini berdampak pada bagaimana volume pod yang dihentikan ditangani oleh kubelet. Jika Anda menggunakan flag ini dalam konfigurasi node Anda, Anda harus memperbarui skrip bootstrap Anda dan meluncurkan template untuk menghapusnya sebelum memutakhirkan.
- Gerbang `VolumeAttributesClass` fitur beta dan sumber daya API diaktifkan dalam versi Amazon EKS 1.31. Fitur ini memungkinkan operator kluster untuk memodifikasi properti yang dapat berubah dari Persistent Volume (PVs) yang dikelola oleh Driver CSI yang kompatibel, termasuk Driver Amazon EBS CSI. Untuk memanfaatkan fitur ini, pastikan Driver CSI Anda mendukung `VolumeAttributesClass` fitur tersebut (untuk Driver Amazon EBS CSI, tingkatkan ke versi 1.35.0 atau yang lebih baru untuk mengaktifkan fitur secara otomatis). Anda akan dapat membuat `VolumeAttributesClass` objek untuk menentukan atribut volume yang diinginkan, seperti jenis volume dan throughput, dan mengaitkannya dengan Klaim Volume Persisten (PVCs). Lihat dokumentasi [resmi Kubernetes serta dokumentasi](#) driver CSI Anda untuk informasi lebih lanjut.
- Untuk informasi selengkapnya tentang Driver Amazon EBS CSI, lihat [the section called "Amazon EBS"](#).
- Dukungan Kubernetes untuk [AppArmor](#) telah lulus menjadi stabil dan sekarang umumnya tersedia untuk penggunaan umum. Fitur ini memungkinkan Anda untuk melindungi wadah Anda AppArmor dengan mengatur `appArmorProfile.type` bidang di `wadahsecurityContext`. Sebelum versi Kubernetes 1.30, dikendalikan AppArmor oleh anotasi. Dimulai dengan versi 1.30, dikontrol

menggunakan bidang. Untuk memanfaatkan fitur ini, kami sarankan untuk bermigrasi dari anotasi dan menggunakan `appArmorProfile.type` bidang untuk memastikan bahwa beban kerja Anda kompatibel.

- Fitur waktu transisi fase `PersistentVolume` terakhir telah lulus menjadi stabil dan sekarang umumnya tersedia untuk penggunaan publik dalam versi Kubernetes. 1.31 Fitur ini memperkenalkan bidang baru, di `.status.lastTransitionTime PersistentVolumeStatus`, yang menyediakan stempel waktu kapan `PersistentVolume` terakhir dialihkan ke fase yang berbeda. Peningkatan ini memungkinkan pelacakan dan pengelolaan yang lebih baik `PersistentVolumes`, terutama dalam skenario di mana memahami siklus hidup volume adalah penting.

Untuk 1.31 changelog Kubernetes yang lengkap, lihat -1.31.md <https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG>

Kubernetes 1.30

Kubernetes sekarang 1.30 tersedia di Amazon EKS. Untuk informasi lebih lanjut tentang Kubernetes 1.30, lihat pengumuman rilis [resmi](#).

Important

- Dimulai dengan versi Amazon EKS 1.30 atau yang lebih baru, grup node terkelola yang baru dibuat akan secara otomatis default menggunakan Amazon Linux 2023 (AL2023) sebagai sistem operasi node. Sebelumnya, grup node baru akan default ke Amazon Linux 2 (AL2). Anda dapat terus menggunakannya AL2 dengan memilihnya sebagai tipe AMI saat membuat grup node baru.
 - Untuk informasi tentang migrasi dari AL2 AL2 023, lihat. [the section called “Tingkatkan ke AL2 023”](#)
 - Untuk informasi selengkapnya tentang Amazon Linux, lihat [Membandingkan AL2 dan AL2 023](#) di Panduan Pengguna Amazon Linux.
 - Untuk informasi selengkapnya tentang menentukan sistem operasi untuk grup node terkelola, lihat. [the section called “Buat”](#)
-
- Dengan Amazon EKS 1.30, `topology.k8s.aws/zone-id` label ditambahkan ke node pekerja. Anda dapat menggunakan Availability Zone IDs (AZ IDs) untuk menentukan lokasi sumber daya

dalam satu akun relatif terhadap sumber daya di akun lain. Untuk informasi selengkapnya, lihat [Availability Zone IDs untuk AWS sumber daya Anda](#) di Panduan Pengguna AWS IAM.

- Dimulai dengan 1.30, Amazon EKS tidak lagi menyertakan default anotasi pada gp2 StorageClass sumber daya yang diterapkan ke cluster yang baru dibuat. Ini tidak berdampak jika Anda mereferensikan kelas penyimpanan ini dengan nama. Anda harus mengambil tindakan jika Anda mengandalkan memiliki default StorageClass di cluster. Anda harus StorageClass mereferensikan namanya gp2. Atau, Anda dapat menerapkan kelas penyimpanan default yang direkomendasikan Amazon EBS dengan menyetel defaultStorageClass.enabled parameter ke true saat menginstal versi 1.31.0 atau yang lebih baru. `aws-ebs-csi-driver` add-on
- Kebijakan IAM minimum yang diperlukan untuk peran IAM cluster Amazon EKS telah berubah. Tindakan `ec2:DescribeAvailabilityZones` itu diperlukan. Untuk informasi selengkapnya, lihat [the section called "IAM role kluster"](#).

Untuk 1.30 changelog Kubernetes selengkapnya, lihat [-1.30.md](#). <https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG>

Tinjau catatan rilis untuk versi Kubernetes pada dukungan yang diperluas

Amazon EKS mendukung versi Kubernetes lebih lama dari yang didukung upstream, dengan dukungan standar untuk versi minor Kubernetes selama 14 bulan sejak dirilis di Amazon EKS, dan dukungan tambahan untuk versi minor Kubernetes untuk dukungan tambahan 12 bulan (total 26 bulan per versi).

Topik ini memberikan perubahan penting yang harus diperhatikan untuk setiap Kubernetes versi dalam dukungan yang diperluas. Saat memutakhirkan, tinjau dengan cermat perubahan yang terjadi antara versi lama dan baru untuk kluster Anda.

Kubernetes 1.29

Kubernetes sekarang 1.29 tersedia di Amazon EKS. Untuk informasi lebih lanjut tentang Kubernetes 1.29, lihat pengumuman rilis [resmi](#).

Important

- Versi `flowcontrol.apiserver.k8s.io/v1beta2` API yang tidak `PriorityLevelConfiguration` digunakan lagi dari `FlowSchema` dan tidak lagi disajikan dalam versi Kubernetes. 1.29 Jika Anda memiliki manifes atau perangkat

lunak klien yang menggunakan grup API beta yang tidak digunakan lagi, Anda harus mengubahnya sebelum memutakhirkan ke versi. 1.29

- `.status.kubeProxyVersion` bidang untuk objek node sekarang tidak digunakan lagi, dan proyek Kubernetes mengusulkan untuk menghapus bidang itu di rilis mendatang. Bidang usang tidak akurat dan secara historis telah dikelola oleh `kubelet` - yang sebenarnya tidak mengetahui `kube-proxy` versinya, atau bahkan apakah sedang berjalan. `kube-proxy` Jika Anda telah menggunakan bidang ini dalam perangkat lunak klien, hentikan - informasinya tidak dapat diandalkan dan bidang tersebut sekarang tidak digunakan lagi.
- Di Kubernetes 1.29 untuk mengurangi potensi permukaan serangan, `LegacyServiceAccountTokenCleanUp` fitur tersebut memberi label token berbasis rahasia yang dibuat secara otomatis sebagai tidak valid jika tidak digunakan untuk waktu yang lama (1 tahun secara default), dan secara otomatis menghapusnya jika penggunaan tidak dicoba untuk waktu yang lama setelah ditandai sebagai tidak valid (1 tahun tambahan secara default). Untuk mengidentifikasi token tersebut, Anda dapat menjalankan:

```
kubectl get cm kube-apiserver-legacy-service-account-token-tracking -n kube-system
```

Untuk 1.29 changelog Kubernetes yang lengkap, lihat [-1.29.md# 1280. https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.29.md#1280](https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.29.md#1280).

Kubernetes 1.28

Kubernetes sekarang 1.28 tersedia di Amazon EKS. Untuk informasi lebih lanjut tentang Kubernetes 1.28, lihat pengumuman rilis [resmi](#).

- Kubernetes v1.28 memperluas kemiringan yang didukung antara node inti dan komponen bidang kontrol dengan satu versi minor, dari `n-2` ke `n-3`, sehingga komponen node (`kubelet` dan `kube-proxy`) untuk versi minor tertua yang didukung dapat bekerja dengan komponen bidang kontrol (`kube-apiserver`, `kube-scheduler`, `kube-controller-manager`, `cloud-controller-manager`) untuk versi minor terbaru yang didukung.
- Metrik `force_delete_pods_total` dan `force_delete_pod_errors_total` di Pod GC Controller dalamnya ditingkatkan untuk memperhitungkan semua penghapusan Pod yang kuat. Sebuah alasan ditambahkan ke metrik untuk menunjukkan apakah pod dihapus secara paksa

karena dihentikan, yatim piatu, diakhiri dengan out-of-service taint, atau terminating dan tidak terjadwal.

- `PersistentVolume` (PV) Pengontrol telah dimodifikasi untuk secara otomatis menetapkan default `StorageClass` ke unbound apa pun `PersistentVolumeClaim` dengan yang `storageClassName` tidak disetel. Selain itu, mekanisme validasi `PersistentVolumeClaim` penerimaan dalam server API telah disesuaikan untuk memungkinkan perubahan nilai dari status yang tidak disetel ke nama sebenarnya `StorageClass`.

Untuk 1.28 changelog Kubernetes yang lengkap, lihat -1.28.md# 1270. <https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-changelog-since-v1.28.md>

Kubernetes 1.27

Kubernetes sekarang 1.27 tersedia di Amazon EKS. Untuk informasi lebih lanjut tentang Kubernetes 1.27, lihat pengumuman rilis [resmi](#).

Important

- Dukungan untuk anotasi `seccomp.security.alpha.kubernetes.io/pod` dan `seccomp` anotasi alfa telah container.`seccomp.security.alpha.kubernetes.io` dihapus. `seccomp` Anotasi alfa tidak digunakan lagi 1.19, dan dengan penghapusannya di 1.27, `seccomp` bidang tidak akan lagi terisi otomatis dengan anotasi. Pods `seccomp` Sebagai gantinya, gunakan `securityContext.seccompProfile` bidang untuk Pods atau wadah untuk mengonfigurasi `seccomp` profil. Untuk memeriksa apakah Anda menggunakan `seccomp` anotasi alfa yang tidak digunakan lagi di kluster Anda, jalankan perintah berikut:

```
kubectl get pods --all-namespaces -o json | grep
-E 'seccomp.security.alpha.kubernetes.io/pod|
container.seccomp.security.alpha.kubernetes.io'
```

- Argumen baris `--container-runtime` perintah untuk kubelet telah dihapus. Runtime kontainer default untuk Amazon EKS telah ada `containerd` sejak saat itu 1.24, yang menghilangkan kebutuhan untuk menentukan runtime kontainer. Dari 1.27 dan seterusnya, Amazon EKS akan mengabaikan `--container-runtime` argumen yang diteruskan ke skrip bootstrap apa pun. Penting agar Anda tidak meneruskan argumen ini untuk `--kubelet-extra-args` mencegah kesalahan selama proses bootstrap

node. Anda harus menghapus `--container-runtime` argumen dari semua alur kerja pembuatan node dan membangun skrip.

- The kubelet in Kubernetes 1.27 meningkatkan default ke dan kubeAPIQPS ke50. kubeAPIBurst 100 Penyempurnaan ini memungkinkan kubelet untuk menangani volume kueri API yang lebih tinggi, meningkatkan waktu respons dan kinerja. Ketika tuntutan untuk Pods meningkat, karena persyaratan penskalaan, default yang direvisi memastikan bahwa secara efisien kubelet dapat mengelola beban kerja yang meningkat. Akibatnya, Pod peluncuran lebih cepat dan operasi cluster lebih efektif.
- Anda dapat menggunakan Pod topologi berbutir lebih halus untuk menyebarkan kebijakan seperti. minDomain Parameter ini memberi Anda kemampuan untuk menentukan jumlah minimum domain yang Pods harus Anda sebar. nodeAffinityPolicy dan nodeTaintPolicy memungkinkan tingkat granularitas ekstra dalam mengatur Pod distribusi. Ini sesuai dengan afinitas simpul, noda, dan matchLabelKeys bidang dalam spesifikasi AndatopologySpreadConstraints. Pod's Hal ini memungkinkan pemilihan Pods untuk menyebarkan perhitungan setelah upgrade bergulir.
- Kubernetes 1.27 mempromosikan ke beta mekanisme kebijakan baru untuk StatefulSets itu mengontrol masa pakai () merekaPersistentVolumeClaims. PVCs Kebijakan PVC retensi baru memungkinkan Anda menentukan apakah PVCs yang dihasilkan dari templat StatefulSet spesifikasi akan dihapus atau dipertahankan secara otomatis saat dihapus atau replika di dalamnya StatefulSet diperkecil. StatefulSet
- Opsi [goaway-chance](#) di server API Kubernetes membantu mencegah koneksi HTTP/2 klien macet pada satu instance server API, dengan menutup koneksi secara acak. Ketika koneksi ditutup, klien akan mencoba untuk menyambung kembali, dan kemungkinan akan mendarat di server API yang berbeda sebagai akibat dari load balancing. Versi Amazon EKS 1.27 telah mengaktifkan goaway-chance bendera. Jika beban kerja Anda yang berjalan di klaster Amazon EKS menggunakan klien yang tidak kompatibel dengan [HTTP GOAWAY](#), kami sarankan Anda memperbarui klien untuk ditangani GOAWAY dengan menghubungkan kembali saat penghentian koneksi.

[Untuk 1.27 changelog Kubernetes yang lengkap, lihat -1.27.md# 1260. <https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-changelog-since-v>](#)

Lihat periode dukungan klaster saat ini

Bagian periode dukungan klaster AWS konsol menunjukkan apakah klaster Anda saat ini menggunakan dukungan standar atau diperpanjang. Jika periode dukungan klaster Anda adalah Dukungan yang diperpanjang, Anda akan dikenakan biaya untuk dukungan yang diperpanjang EKS.

Untuk informasi selengkapnya tentang dukungan standar dan diperpanjang, lihat [the section called “Versi Kubernetes”](#).

1. Arahkan ke halaman Clusters di bagian EKS AWS Konsol. Konfirmasikan bahwa konsol disetel ke AWS wilayah yang sama dengan cluster yang ingin Anda tinjau.
2. Tinjau kolom Support Period. Jika nilainya adalah dukungan Standar hingga..., Anda saat ini tidak dikenakan biaya untuk dukungan yang diperpanjang. Anda berada dalam periode dukungan standar. Jika nilainya Extended support... cluster ini saat ini sedang dikenakan biaya untuk dukungan yang diperpanjang.

Note

Periode Support tidak dapat diambil dengan AWS API atau CLI.

Lihat kebijakan peningkatan klaster saat ini

Kebijakan upgrade klaster menentukan apa yang terjadi pada klaster Anda ketika meninggalkan periode dukungan standar. Jika kebijakan pemutakhiran Anda EXTENDED, klaster tidak akan ditingkatkan secara otomatis, dan akan masuk ke dukungan yang diperluas. Jika kebijakan pemutakhiran Anda STANDARD, itu akan ditingkatkan secara otomatis.

Kontrol Amazon EKS untuk kebijakan versi Kubernetes memungkinkan Anda memilih akhir dari perilaku dukungan standar untuk kluster EKS Anda. Dengan kontrol ini, Anda dapat memutuskan cluster mana yang harus memasukkan dukungan yang diperluas dan cluster mana yang harus ditingkatkan secara otomatis di akhir dukungan standar untuk versi Kubernetes.

Versi minor berada di bawah dukungan standar di Amazon EKS selama 14 bulan pertama setelah dirilis. Setelah versi melewati akhir tanggal dukungan standar, ia memasuki dukungan diperpanjang untuk 12 bulan ke depan. Dukungan yang diperluas memungkinkan Anda untuk tinggal di versi Kubernetes tertentu lebih lama dengan biaya tambahan per jam cluster. Anda dapat mengaktifkan atau menonaktifkan dukungan tambahan untuk Kluster EKS. Jika Anda menonaktifkan dukungan

diperpanjang, secara otomatis AWS akan meng-upgrade cluster Anda ke versi berikutnya di akhir dukungan standar. Jika Anda mengaktifkan dukungan yang diperpanjang, Anda dapat tetap menggunakan versi saat ini dengan biaya tambahan untuk jangka waktu terbatas. Rencanakan untuk meng-upgrade kluster Kubernetes Anda secara teratur, bahkan jika Anda menggunakan dukungan yang diperluas.

Anda dapat menyetel kebijakan versi untuk kluster baru dan yang sudah ada, menggunakan `supportType` properti. Ada dua opsi yang dapat digunakan untuk mengatur kebijakan dukungan versi:

- **STANDARD** — Kluster EKS Anda memenuhi syarat untuk peningkatan otomatis di akhir dukungan standar. Anda tidak akan dikenakan biaya dukungan tambahan dengan pengaturan ini tetapi kluster EKS Anda akan secara otomatis meningkatkan ke versi Kubernetes yang didukung berikutnya dalam dukungan standar.
- **EXTENDED** — Kluster EKS Anda akan masuk ke dukungan yang diperluas setelah versi Kubernetes mencapai akhir dukungan standar. Anda akan dikenakan biaya dukungan yang diperpanjang dengan pengaturan ini. Anda dapat meningkatkan kluster Anda ke versi Kubernetes standar yang didukung untuk menghentikan biaya dukungan yang diperpanjang. Cluster yang berjalan pada dukungan yang diperpanjang akan memenuhi syarat untuk peningkatan otomatis di akhir dukungan yang diperpanjang.

Dukungan yang diperluas diaktifkan secara default untuk cluster baru, dan cluster yang ada. Anda dapat melihat apakah dukungan diperpanjang diaktifkan untuk cluster di AWS Management Console, atau dengan menggunakan AWS CLI.

 Important

Jika Anda ingin kluster Anda tetap menggunakan versi Kubernetes saat ini untuk memanfaatkan periode dukungan yang diperpanjang, Anda harus mengaktifkan kebijakan peningkatan dukungan yang diperpanjang sebelum akhir periode dukungan standar.

Anda hanya dapat mengatur kebijakan dukungan versi untuk kluster Anda saat dijalankan pada versi Kubernetes dalam dukungan standar. Setelah versi memasuki dukungan yang diperluas, Anda tidak akan dapat mengubah pengaturan ini sampai Anda menjalankan versi dalam dukungan standar.

Misalnya, jika Anda telah menetapkan kebijakan dukungan versi seperti `standard` itu, Anda tidak akan dapat mengubah pengaturan ini setelah versi Kubernetes yang berjalan di kluster Anda

mencapai akhir dukungan standar. Jika Anda telah menetapkan kebijakan dukungan versi Anda, `extended` maka Anda tidak akan dapat mengubah pengaturan ini setelah versi Kubernetes yang berjalan di klaster Anda mencapai akhir dukungan standar. Untuk mengubah pengaturan kebijakan dukungan versi, klaster Anda harus berjalan pada versi Kubernetes standar yang didukung.

Lihat kebijakan peningkatan klaster (AWS Konsol)

1. Arahkan ke halaman Clusters di bagian EKS AWS Konsol. Konfirmasikan bahwa konsol disetel ke AWS wilayah yang sama dengan cluster yang ingin Anda tinjau.
2. Tinjau kolom Kebijakan Peningkatan. Jika nilainya `Standard Support`, klaster Anda tidak akan memasukkan dukungan yang diperluas. Jika nilainya `Extended Support`, klaster Anda akan memasukkan dukungan yang diperluas.

Lihat kebijakan peningkatan klaster (AWS CLI)

1. Verifikasi AWS CLI diinstal dan Anda masuk. [Pelajari cara memperbarui dan menginstal AWS CLI.](#)
2. Tentukan nama cluster EKS Anda. Atur CLI ke AWS wilayah yang sama dengan cluster EKS Anda.
3. Jalankan perintah berikut:

```
aws eks describe-cluster \
  --name <cluster-name> \
  --query "cluster.upgradePolicy.supportType"
```

4. Jika nilainya `STANDARD`, klaster Anda tidak akan memasukkan dukungan yang diperluas. Jika nilainya `EXTENDED`, klaster Anda akan memasukkan dukungan yang diperluas.

Tambahkan fleksibilitas untuk merencanakan peningkatan versi Kubernetes dengan mengaktifkan dukungan EKS yang diperluas

Topik ini menjelaskan cara menyetel kebijakan pemutakhiran klaster EKS untuk mengaktifkan dukungan yang diperluas. Kebijakan upgrade klaster EKS menentukan apa yang terjadi ketika klaster mencapai akhir periode dukungan standar. Jika kebijakan peningkatan klaster telah mengaktifkan dukungan tambahan, kebijakan tersebut akan memasuki periode dukungan yang diperpanjang pada akhir periode dukungan standar. Cluster tidak akan ditingkatkan secara otomatis pada akhir periode dukungan standar.

Cluster sebenarnya dalam periode dukungan yang diperpanjang menimbulkan biaya yang lebih tinggi. Jika kluster hanya memiliki kebijakan peningkatan yang ditetapkan untuk mengaktifkan dukungan yang diperpanjang, dan sebaliknya dalam periode dukungan standar, itu menimbulkan biaya standar.

Jika Anda membuat kluster di AWS konsol, kebijakan pemutakhiran akan disetel untuk menonaktifkan dukungan yang diperluas. Jika Anda membuat kluster dengan cara lain, kebijakan pemutakhiran akan disetel untuk mengaktifkan dukungan yang diperluas. Misalnya, cluster yang dibuat dengan AWS API telah mengaktifkan dukungan tambahan.

Untuk informasi selengkapnya tentang kebijakan peningkatan, lihat [the section called “Lihat kebijakan peningkatan”](#).

Important

Jika Anda ingin kluster Anda tetap menggunakan versi Kubernetes saat ini untuk memanfaatkan periode dukungan yang diperpanjang, Anda harus mengaktifkan kebijakan peningkatan dukungan yang diperpanjang sebelum akhir periode dukungan standar. Jika Anda tidak mengaktifkan dukungan yang diperluas, kluster Anda akan ditingkatkan secara otomatis.

Aktifkan dukungan tambahan EKS (AWS Konsol)

1. Arahkan ke kluster EKS Anda di AWS Konsol. Pilih tab Ikhtisar pada halaman Info Cluster.
2. Di bagian pengaturan versi Kubernetes, pilih Kelola.
3. Pilih Dukungan diperpanjang dan kemudian Simpan perubahan.

Aktifkan dukungan diperpanjang EKS (AWS CLI)

1. Verifikasi AWS CLI diinstal dan Anda masuk. [Pelajari cara memperbarui dan menginstal AWS CLI.](#)
2. Tentukan nama cluster EKS Anda.
3. Jalankan perintah berikut:

```
aws eks update-cluster-config \
--name <cluster-name> \
--upgrade-policy supportType=EXTENDED
```

Cegah peningkatan biaya cluster dengan menonaktifkan dukungan EKS yang diperluas

Topik ini menjelaskan cara menyetel kebijakan pemutakhiran kluster EKS untuk menonaktifkan dukungan yang diperluas. Kebijakan pemutakhiran kluster EKS menentukan apa yang terjadi ketika kluster mencapai akhir periode dukungan standar. Jika kebijakan pemutakhiran kluster telah menonaktifkan dukungan, maka secara otomatis akan ditingkatkan ke versi Kubernetes berikutnya.

Untuk informasi selengkapnya tentang kebijakan peningkatan, lihat [the section called “Lihat kebijakan peningkatan”](#).

Important

Anda tidak dapat menonaktifkan dukungan yang diperluas setelah kluster Anda memasukkannya. Anda hanya dapat menonaktifkan dukungan tambahan untuk cluster pada dukungan standar.

AWS merekomendasikan untuk meningkatkan kluster Anda ke versi dalam periode dukungan standar.

Nonaktifkan dukungan diperpanjang EKS (AWS Konsol)

1. Arahkan ke kluster EKS Anda di AWS Konsol. Pilih tab Ikhtisar pada halaman Info Cluster.
2. Di bagian pengaturan versi Kubernetes, pilih Kelola.
3. Pilih Dukungan standar dan kemudian Simpan perubahan.

Nonaktifkan dukungan diperpanjang EKS (AWS CLI)

1. Verifikasi AWS CLI diinstal dan Anda masuk. [Pelajari cara memperbarui dan menginstal AWS CLI.](#)
2. Tentukan nama cluster EKS Anda.
3. Jalankan perintah berikut:

```
aws eks update-cluster-config \
--name <cluster-name> \
--upgrade-policy supportType=STANDARD
```


Lihat versi platform Amazon EKS untuk setiap versi Kubernetes

Versi platform Amazon EKS mewakili kemampuan bidang kontrol cluster Amazon EKS, seperti flag server Kubernetes API yang diaktifkan, serta versi patch Kubernetes saat ini. Setiap Kubernetes versi minor memiliki satu atau lebih Amazon EKS versi platform yang terkait. Versi platform untuk versi minor Kubernetes yang berbeda bersifat independen. Anda dapat [mengambil versi platform cluster Anda saat ini](#) menggunakan AWS Management Console CLI atau. Jika Anda memiliki cluster lokal di AWS Outposts, lihat [the section called “Versi platform EKS”](#) alih-alih topik ini.

Ketika versi minor Kubernetes baru tersedia di Amazon EKS, seperti 1.33, versi platform Amazon EKS awal untuk versi minor Kubernetes dimulai pada. eks . 1 Namun, Amazon EKS merilis versi platform baru secara berkala guna mengaktifkan pengaturan pesawat kendali Kubernetes baru dan untuk menyediakan perbaikan keamanan.

Saat versi platform Amazon EKS baru tersedia untuk versi minor:

- Nomor versi platform Amazon EKS bertambah (eks . <n+1>).
- Amazon EKS secara otomatis memutakhirkan semua klaster yang sudah ada ke versi platform Amazon EKS terbaru untuk versi minor Kubernetes yang sesuai. Pemutakhiran otomatis versi platform Amazon EKS yang sudah ada diluncurkan secara bertahap. Proses peluncuran mungkin memakan waktu lama. Jika Anda memerlukan fitur versi platform Amazon EKS terbaru segera, Anda harus membuat klaster Amazon EKS baru.

Jika klaster Anda memiliki lebih dari dua versi platform di belakang versi platform saat ini, kemungkinan Amazon EKS tidak dapat memperbarui cluster Anda secara otomatis. Untuk detail tentang apa yang dapat menyebabkan ini, lihat [the section called “Versi platform Amazon EKS lebih dari dua versi di belakang versi platform saat ini”](#).

- Amazon EKS boleh menerbitkan AMI simpul baru dengan versi patch yang sesuai. Namun, semua versi patch kompatibel antara bidang kontrol EKS dan node AMIs untuk versi minor Kubernetes tertentu.

Versi platform Amazon EKS baru tidak memperkenalkan perubahan yang melanggar atau menyebabkan gangguan layanan.

Klaster selalu dibuat dengan versi platform Amazon EKS terbaru yang tersedia (eks . <n>) untuk versi Kubernetes yang ditentukan. Jika Anda memperbarui klaster ke versi minor Kubernetes baru, klaster Anda menerima versi platform Amazon EKS saat ini ke versi minor Kubernetes yang telah Anda perbarui.

Versi platform Amazon EKS saat ini dan terbaru dideskripsikan dalam tabel berikut.

 **Note**

AWS baru-baru ini menonaktifkan beberapa versi platform yang diterbitkan pada Juni 2024. Versi platform memiliki masalah stabilitas. Tidak ada tindakan yang diperlukan.

Untuk menerima pemberitahuan semua perubahan file sumber ke halaman dokumentasi khusus ini, Anda dapat berlangganan URL berikut dengan pembaca RSS:

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/clusters/platform-versions.adoc.atom
```

Versi Kubernetes 1.33

Pengontrol penerimaan berikut diaktifkan untuk semua versi 1.33
platform:NodeRestriction,,ExtendedResourceToleration,
NamespaceLifecycleLimitRanger,ServiceAccount,TaintNodesByCondition,PodSecurity,Pri
MutatingAdmissionWebhookValidatingAdmissionWebhook,ResourceQuota.

Versi Kubernetes	Versi platform EKS	Catatan rilis	Tanggal rilis
1.33.1	eks.6	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Juni 26, 2025
1.33.1	eks.5	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Juni 11, 2025
1.33.1	eks.4	Rilis awal versi 1.33 Kubernetes untuk EKS. Untuk informasi selengkapnya, lihat	28 Mei 2025

Versi Kubernetes	Versi platform EKS	Catatan rilis	Tanggal rilis
		the section called “Kubernetes 1.33” .	

Versi Kubernetes 1.32

Pengontrol penerimaan berikut diaktifkan untuk semua versi 1.32

platform:NodeRestriction,,,ExtendedResourceToleration,

NamespaceLifecycleLimitRanger,ServiceAccount,TaintNodesByCondition,PodSecurity,Pri

MutatingAdmissionWebhookValidatingAdmissionWebhook,ResourceQuota.

Versi Kubernetes	Versi platform EKS	Catatan rilis	Tanggal rilis
1.32.5	eks.13	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Juni 26, 2025
1.32.5	eks.12	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Juni 11, 2025
1.32.5	eks.11	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	30 Mei 2025
1.32.3	eks.10	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	16 Mei 2025
1.32.3	eks.9	Versi platform baru dengan perbaikan	April 29, 2025

Versi Kubernetes	Versi platform EKS	Catatan rilis	Tanggal rilis
		dan penyempurnaan keamanan.	
1.32.3	eks.8	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	April 18, 2025
1.32.3	eks.7	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	April 18, 2025
1.32.3	eks.6	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	April 2, 2025
1.32.2	eks.5	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Maret 17, 2025
1.32.2	eks.4	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Maret 4, 2025
1.32.1	eks.3	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Februari 24, 2025

Versi Kubernetes	Versi platform EKS	Catatan rilis	Tanggal rilis
1.32.0	eks.2	Rilis awal versi 1.32 Kubernetes untuk EKS. Untuk informasi selengkapnya, lihat the section called “Kubernetes 1.32” .	Januari 2025

Versi Kubernetes 1.31

Pengontrol penerimaan berikut diaktifkan untuk semua versi 1.31

platform:NodeRestriction,,,ExtendedResourceToleration,

NamespaceLifecycleLimitRanger,ServiceAccount,TaintNodesByCondition,PodSecurity,Pri

ValidatingAdmissionWebhookResourceQuota,ObjectCount.

Versi Kubernetes	Versi platform EKS	Catatan rilis	Tanggal rilis
1.31.9	eks.29	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Juni 26, 2025
1.31.9	eks.28	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Juni 11, 2025
1.31.9	eks.27	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	30 Mei 2025
1.31.7	eks.26	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	16 Mei 2025

Versi Kubernetes	Versi platform EKS	Catatan rilis	Tanggal rilis
1.31.7	eks.25	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	April 29, 2025
1.31.7	eks.24	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	April 18, 2025
1.31.7	eks.23	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	April 18, 2025
1.31.7	eks.22	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	April 2, 2025
1.31.6	eks.21	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Maret 17, 2025
1.31.6	eks.20	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Maret 4, 2025
1.31.5	eks.19	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Februari 24, 2025

Versi Kubernetes	Versi platform EKS	Catatan rilis	Tanggal rilis
1.31.5	eks.18	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Februari 24, 2025
1.31.4	eks.17	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Januari 17, 2025
1.31.2	eks.12	Versi platform baru dengan Amazon EKS Hybrid Nodes mendukung dan penyempurnaan untuk mengontro l observabilitas pesawat. Lihat the section called “Node hibrida” dan lihat Amazon EKS masing-masing meningkatkan observabilitas kinerja .	November 15, 2024
1.31.1	eks.6	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Oktober 21, 2024
1.31.0	eks.4	Rilis awal versi 1.31 Kubernetes untuk EKS. Untuk informasi selengkapnya, lihat the section called “Kubernetes 1.31” .	September 26, 2024

Versi Kubernetes 1.30

Pengontrol penerimaan berikut diaktifkan untuk semua versi 1.30

platform:NodeRestriction,,,ExtendedResourceToleration,

NamespaceLifecycleLimitRanger,ServiceAccount,TaintNodesByCondition,PodSecurity,Pri

MutatingAdmissionWebhookValidatingAdmissionWebhook,ResourceQuota.

Versi Kubernetes	Versi platform EKS	Catatan rilis	Tanggal rilis
1.30.13	eks.37	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Juni 26, 2025
1.30.13	eks.36	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Juni 11, 2025
1.30.13	eks.35	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	30 Mei 2025
1.30.11	eks.34	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	16 Mei 2025
1.30.11	eks.33	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	April 29, 2025
1.30.11	eks.32	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	April 18, 2025

Versi Kubernetes	Versi platform EKS	Catatan rilis	Tanggal rilis
1.30.11	eks.31	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	April 18, 2025
1.30.11	eks.30	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	April 2, 2025
1.30.10	eks.29	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Maret 17, 2025
1.30.10	eks.28	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Maret 4, 2025
1.30.9	eks.27	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Februari 24, 2025
1.30.8	eks.25	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Januari 17, 2025
1.30.8	eks.24	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Januari 3, 2025

Versi Kubernetes	Versi platform EKS	Catatan rilis	Tanggal rilis
1.30.7	eks.23	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Desember 13, 2024
1.30.6	eks.22	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Desember 13, 2024
1.30.6	eks.21	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Desember 13, 2024
1.30.6	eks.20	Versi platform baru dengan Amazon EKS Hybrid Nodes mendukung dan penyempurnaan untuk mengontro l observabilitas pesawat. Lihat the section called “Node hibrida” dan lihat Amazon EKS masing-masing meningkatkan observabilitas kinerja.	November 15, 2024
1.30.5	eks.12	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Oktober 21, 2024

Versi Kubernetes	Versi platform EKS	Catatan rilis	Tanggal rilis
1.30.4	eks.8	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	September 3, 2024
1.30.3	eks.7	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Agustus 28, 2024
1.30.3	eks.6	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Agustus 9, 2024
1.30.2	eks.5	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Juli 2, 2024
1.30.0	eks.2	Rilis awal versi 1.30 Kubernetes untuk EKS. Untuk informasi selengkapnya, lihat the section called “Kubernetes 1.30” .	Mei 23, 2024

Versi Kubernetes 1.29

Pengontrol penerimaan berikut diaktifkan untuk semua versi 1.29

platform:NodeRestriction,,,ExtendedResourceToleration,

NamespaceLifecycleLimitRanger,ServiceAccount,TaintNodesByCondition,PodSecurity,Pri

MutatingAdmissionWebhookValidatingAdmissionWebhook,ResourceQuota.

Versi Kubernetes	Versi platform EKS	Catatan rilis	Tanggal rilis
1.29.15	eks.40	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Juni 26, 2025
1.29.15	eks.39	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Juni 11, 2025
1.29.15	eks.38	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	30 Mei 2025
1.29.15	eks.37	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	16 Mei 2025
1.29.15	eks.36	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	April 29, 2025
1.29.15	eks.35	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	April 18, 2025
1.29.15	eks.34	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	April 18.2025

Versi Kubernetes	Versi platform EKS	Catatan rilis	Tanggal rilis
1.29.15	eks.33	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	April 2, 2025
1.29.14	eks.32	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Maret 17, 2025
1.29.14	eks.31	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Maret 4, 2025
1.29.13	eks.30	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Februari 24, 2025
1.29.13	eks.29	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Februari 24, 2025
1.29.12	eks.28	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Januari 20, 2025
1.29.12	eks.27	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Januari 3, 2025

Versi Kubernetes	Versi platform EKS	Catatan rilis	Tanggal rilis
1.29.11	eks.26	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Desember 13, 2024
1.29.10	eks.25	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Desember 13, 2024
1.29.10	eks.24	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Desember 13, 2024
1.29.10	eks.23	Versi platform baru dengan Amazon EKS Hybrid Nodes mendukung dan penyempurnaan untuk mengontro l observabilitas pesawat. Lihat the section called “Node hibrida” dan lihat Amazon EKS masing-masing meningkatkan observabilitas kinerja.	November 15, 2024
1.29.9	eks.17	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Oktober 21, 2024

Versi Kubernetes	Versi platform EKS	Catatan rilis	Tanggal rilis
1.29.8	eks.13	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	September 3, 2024
1.29.7	eks.12	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Agustus 28, 2024
1.29.7	eks.11	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Agustus 9, 2024
1.29.6	eks.10	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Juli 2, 2024
1.29.4	eks.7	Versi platform baru dengan penskalaan otomatis CoreDNS, perbaikan keamanan, dan penyempurnaan. Untuk informasi selengkapnya tentang penskalaan otomatis CoreDNS, lihat. the section called “Skala untuk lalu lintas tinggi”	16 Mei 2024
1.29.3	eks.6	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	April 18, 2024

Versi Kubernetes	Versi platform EKS	Catatan rilis	Tanggal rilis
1.29.1	eks.5	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Maret 29, 2024
1.29.1	eks.4	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Maret 20, 2024
1.29.1	eks.3	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Maret 12, 2024
1.29.0	eks.1	Rilis awal versi 1.29 Kubernetes untuk EKS. Untuk informasi selengkapnya, lihat the section called “Kubernetes 1.29” .	23 Januari 2024

Versi Kubernetes 1.28

Pengontrol penerimaan berikut diaktifkan untuk semua versi 1.28

platform:NodeRestriction,,,ExtendedResourceToleration,

NamespaceLifecycleLimitRanger,ServiceAccount,TaintNodesByCondition,PodSecurity,Pri

MutatingAdmissionWebhookValidatingAdmissionWebhook,ResourceQuota.

Versi Kubernetes	Versi platform EKS	Catatan rilis	Tanggal rilis
1.28.15	eks.46	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Juni 26, 2025

Versi Kubernetes	Versi platform EKS	Catatan rilis	Tanggal rilis
1.28.15	eks.45	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Juni 11, 2025
1.28.15	eks.44	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	30 Mei. 2025
1.28.15	eks.43	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	16 Mei 2025
1.28.15	eks.42	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	April 29, 2025
1.28.15	eks.41	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	April 18, 2025
1.28.15	eks.40	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	April 18, 2025
1.28.15	eks.39	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	April 2, 2025

Versi Kubernetes	Versi platform EKS	Catatan rilis	Tanggal rilis
1.28.15	eks.38	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Maret 17, 2025
1.28.15	eks.37	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Maret 4, 2025
1.28.15	eks.36	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Februari 24, 2025
1.28.15	eks.34	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Januari 17, 2025
1.28.15	eks.33	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Januari 3, 2025
1.28.15	eks.32	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Desember 13, 2024
1.28.15	eks.31	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Desember 13, 2024

Versi Kubernetes	Versi platform EKS	Catatan rilis	Tanggal rilis
1.28.15	eks.30	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Desember 13, 2024
1.28.15	eks.29	Versi platform baru dengan Amazon EKS Hybrid Nodes mendukung dan penyempurnaan untuk mengontro l observabilitas pesawat. Lihat the section called “Node hibrida” dan lihat Amazon EKS masing-masing meningkatkan observabilitas kinerja.	November 15, 2024
1.28.14	eks.23	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Oktober 21, 2024
1.28.13	eks.19	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	September 3, 2024
1.28.12	eks.18	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Agustus 28, 2024

Versi Kubernetes	Versi platform EKS	Catatan rilis	Tanggal rilis
1.28.11	eks.17	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Agustus 9, 2024
1.28.11	eks.16	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Juli 2, 2024
1.28.9	eks.13	Versi platform baru dengan penskalaan otomatis CoreDNS, perbaikan keamanan, dan penyempurnaan. Untuk informasi selengkapnya tentang penskalaan otomatis CoreDNS, lihat. the section called “Skala untuk lalu lintas tinggi”	16 Mei 2024
1.28.8	eks.12	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	April 18, 2024
1.28.7	eks.11	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Maret 29, 2024
1.28.7	eks.10	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Maret 20, 2024

Versi Kubernetes	Versi platform EKS	Catatan rilis	Tanggal rilis
1.28.6	eks.9	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Maret 12, 2024
1.28.5	eks.7	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Januari 17, 2024
1.28.4	eks.6	Versi platform baru dengan entri akses , perbaikan keamanan, dan penyempurnaan.	14 Desember 2023
1.28.4	eks.5	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Desember 12, 2023
1.28.3	eks.4	Versi platform baru dengan Pelajari cara EKS Pod Identity memberikan Pod akses ke AWS layanan , perbaikan keamanan, dan penyempurnaan.	10 November 2023
1.28.3	eks.3	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	3 November 2023

Versi Kubernetes	Versi platform EKS	Catatan rilis	Tanggal rilis
1.28.2	eks.2	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	16 Oktober 2023
1.28.1	eks.1	Rilis awal versi 1.28 Kubernetes untuk EKS. Untuk informasi selengkapnya, lihat the section called “Kubernetes 1.28” .	26 September 2023

Versi Kubernetes 1.27

Pengontrol penerimaan berikut diaktifkan untuk semua versi 1.27

platform:NodeRestriction,,,ExtendedResourceToleration,

NamespaceLifecycleLimitRanger,ServiceAccount,TaintNodesByCondition,PodSecurity,Pri

MutatingAdmissionWebhookValidatingAdmissionWebhook,ResourceQuota.

Versi Kubernetes	Versi platform EKS	Catatan rilis	Tanggal rilis
1.27.16	eks.50	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Juni 26, 2025
1.27.16	eks.49	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Juni 11, 2025
1.27.16	eks.48	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	30 Mei 2025

Versi Kubernetes	Versi platform EKS	Catatan rilis	Tanggal rilis
1.27.16	eks.47	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	16 Mei 2025
1.27.16	eks.46	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	April 29, 2025
1.27.16	eks.45	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	April 18, 2025
1.27.16	eks.44	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	April 18, 2025
1.27.16	eks.43	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	April 2, 2025
1.27.16	eks.42	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Maret 17, 2025
1.27.16	eks.41	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Maret 4, 2025

Versi Kubernetes	Versi platform EKS	Catatan rilis	Tanggal rilis
1.27.16	eks.40	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Maret 4, 2025
1.27.16	eks.38	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Januari 17, 2025
1.27.16	eks.37	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Januari 3, 2025
1.27.16	eks.36	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Desember 13, 2024
1.27.16	eks.35	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Desember 13, 2024
1.27.16	eks.34	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Desember 13, 2024

Versi Kubernetes	Versi platform EKS	Catatan rilis	Tanggal rilis
1.27.16	eks.33	Versi platform baru dengan dukungan Amazon EKS Hybrid Nodes, perbaikan keamanan, dan penyempurnaan. Untuk informasi selengkapnya tentang Amazon EKS Hybrid Nodes, lihat the section called “Node hibrida” .	November 15, 2024
1.27.16	eks.27	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Oktober 21, 2024
1.27.16	eks.23	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	September 3, 2024
1.27.16	eks.22	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Agustus 28, 2024
1.27.16	eks.21	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Agustus 9, 2024

Versi Kubernetes	Versi platform EKS	Catatan rilis	Tanggal rilis
1.27.15	eks.20	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Juli 2, 2024
1.27.13	eks.17	Versi platform baru dengan penskalaan otomatis CoreDNS, perbaikan keamanan, dan penyempurnaan. Untuk informasi selengkapnya tentang penskalaan otomatis CoreDNS, lihat the section called “Skala untuk lalu lintas tinggi”	16 Mei 2024
1.27.12	eks.16	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	April 18, 2024
1.27.11	eks.15	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Maret 29, 2024
1.27.11	eks.14	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Maret 20, 2024
1.27.10	eks.13	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Maret 12, 2024

Versi Kubernetes	Versi platform EKS	Catatan rilis	Tanggal rilis
1.27.9	eks.11	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Januari 17, 2024
1.27.8	eks.10	Versi platform baru dengan entri akses , perbaikan keamanan, dan penyempurnaan.	14 Desember 2023
1.27.8	eks.9	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Desember 12, 2023
1.27.7	eks.8	Versi platform baru dengan Pelajari cara EKS Pod Identity memberikan Pod akses ke AWS layanan , perbaikan keamanan, dan penyempurnaan.	10 November 2023
1.27.7	eks.7	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	3 November 2023
1.27.6	eks.6	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	16 Oktober 2023

Versi Kubernetes	Versi platform EKS	Catatan rilis	Tanggal rilis
1.27.4	eks.5	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Agustus 30, 2023
1.27.4	eks.4	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Juli 30, 2023
1.27.3	eks.3	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Juni 30, 2023
1.27.2	eks.2	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	9 Juni 2023
1.27.1	eks.1	Rilis awal versi 1.27 Kubernetes untuk EKS. Untuk informasi selengkapnya, lihat the section called “Kubernetes 1.27” .	24 Mei 2023

Versi Kubernetes 1.26

Pengontrol penerimaan berikut diaktifkan untuk semua versi 1.26

platform:NodeRestriction,,,ExtendedResourceToleration,

NamespaceLifecycleLimitRanger,ServiceAccount,TaintNodesByCondition,PodSecurity,Pri

MutatingAdmissionWebhookValidatingAdmissionWebhook,ResourceQuota.

Versi Kubernetes	Versi platform EKS	Catatan rilis	Tanggal rilis
1.26.15	eks.51	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Juni 11, 2025
1.26.15	eks.50	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	30 Mei 2025
1.26.15	eks.49	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	16 Mei 2025
1.26.15	eks.48	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	April 29, 2025
1.26.15	eks.47	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	April 18, 2025
1.26.15	eks.46	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	April 18, 2025
1.26.15	eks.45	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	April 2, 2025

Versi Kubernetes	Versi platform EKS	Catatan rilis	Tanggal rilis
1.26.15	eks.44	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Maret 17, 2025
1.26.15	eks.43	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Maret 4, 2025
1.26.15	eks.42	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Februari 24, 2025
1.26.15	eks.40	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Januari 17, 2025
1.26.15	eks.39	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Januari 3, 2025
1.26.15	eks.38	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Desember 13, 2024
1.26.15	eks.37	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Desember 13, 2024

Versi Kubernetes	Versi platform EKS	Catatan rilis	Tanggal rilis
1.26.15	eks.36	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Desember 13, 2024
1.26.15	eks.35	Versi platform baru dengan dukungan Amazon EKS Hybrid Nodes, perbaikan keamanan, dan penyempurnaan. Untuk informasi selengkapnya tentang Amazon EKS Hybrid Nodes, lihat the section called “Node hibrida” .	November 15, 2024
1.26.15	eks.28	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Oktober 21, 2024
1.26.15	eks.24	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	September 3, 2024
1.26.15	eks.23	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Agustus 28, 2024

Versi Kubernetes	Versi platform EKS	Catatan rilis	Tanggal rilis
1.26.15	eks.22	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Agustus 9, 2024
1.26.15	eks.21	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Juli 2, 2024
1.26.15	eks.18	Versi platform baru dengan penskalaan otomatis CoreDNS, perbaikan keamanan, dan penyempurnaan. Untuk informasi selengkapnya tentang penskalaan otomatis CoreDNS, lihat the section called “Skala untuk lalu lintas tinggi”	16 Mei 2024
1.26.15	eks.17	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	April 18, 2024
1.26.14	eks.16	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Maret 29, 2024
1.26.14	eks.15	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Maret 20, 2024

Versi Kubernetes	Versi platform EKS	Catatan rilis	Tanggal rilis
1.26.13	eks.14	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Maret 12, 2024
1.26.12	eks.12	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Januari 17, 2024
1.26.11	eks.11	Versi platform baru dengan entri akses , perbaikan keamanan, dan penyempurnaan.	14 Desember 2023
1.26.11	eks.10	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Desember 12, 2023
1.26.10	eks.9	Versi platform baru dengan Pelajari cara EKS Pod Identity memberikan Pod akses ke AWS layanan , perbaikan keamanan, dan penyempurnaan.	10 November 2023
1.26.10	eks.8	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	3 November 2023

Versi Kubernetes	Versi platform EKS	Catatan rilis	Tanggal rilis
1.26.9	eks.7	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	16 Oktober 2023
1.26.7	eks.6	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Agustus 30, 2023
1.26.7	eks.5	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Juli 30, 2023
1.26.6	eks.4	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Juni 30, 2023
1.26.5	eks.3	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	9 Juni 2023
1.26.4	eks.2	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	5 Mei 2023
1.26.2	eks.1	Rilis awal versi 1.26 Kubernetes untuk EKS.	11 April 2023

Dapatkan versi platform saat ini

1. Buka konsol Amazon EKS.
2. Pada panel navigasi, silakan pilih Klaster.
3. Dalam daftar cluster, pilih Nama Cluster untuk memeriksa versi platform.
4. Pilih tab Gambaran Umum.
5. Versi Platform tersedia di bagian Detail.
6. Tentukan Nama cluster yang ingin Anda periksa versi platform.
7. Jalankan perintah berikut:

```
aws eks describe-cluster --name my-cluster --query cluster.platformVersion
```

Contoh output adalah sebagai berikut.

```
"eks.10"
```

Ubah versi platform

Anda tidak dapat mengubah versi platform cluster EKS. Ketika versi platform Amazon EKS baru tersedia untuk versi Kubernetes, EKS secara otomatis meningkatkan semua cluster yang ada ke versi platform Amazon EKS terbaru untuk versi Kubernetes yang sesuai. Pemutakhiran otomatis versi platform Amazon EKS yang sudah ada diluncurkan secara bertahap. Anda tidak dapat menggunakan AWS Konsol atau CLI untuk mengubah versi platform.

Jika Anda meng-upgrade versi Kubernetes Anda, klaster Anda akan beralih ke versi platform terbaru untuk versi Kubernetes.

Komputasi klaster skala dengan Karpenter dan Cluster Autoscaler

Autoscaling adalah fungsi yang secara otomatis mengukur sumber daya Anda keluar dan masuk untuk memenuhi permintaan yang berubah. Ini adalah fungsi utama Kubernetes yang seharusnya membutuhkan sumber daya manusia yang luas untuk bekerja secara manual.

Mode Otomatis EKS

Mode Otomatis Amazon EKS secara otomatis menskalakan sumber daya komputasi kluster. Jika sebuah pod tidak dapat masuk ke node yang ada, Mode Otomatis EKS akan membuat yang baru. Mode Otomatis EKS juga mengkonsolidasikan beban kerja dan menghapus node. Mode Otomatis EKS dibangun di atas Karpenter.

Untuk informasi selengkapnya, lihat:

- [Mode Otomatis EKS](#)
- [the section called “Buat kumpulan simpul”](#)
- [the section called “Menyebarkan beban kerja yang mengembang”](#)

Solusi Tambahan

Amazon EKS mendukung dua produk penskalaan otomatis tambahan:

Karpenter

Karpenter adalah autoscaler cluster Kubernetes yang fleksibel dan berkinerja tinggi yang membantu meningkatkan ketersediaan aplikasi dan efisiensi kluster. Karpenter meluncurkan sumber daya komputasi berukuran tepat (misalnya, EC2 instans Amazon) sebagai respons terhadap perubahan beban aplikasi dalam waktu kurang dari satu menit. Dengan mengintegrasikan Kubernetes dengan Kubernetes AWS, Karpenter dapat menyediakan sumber daya just-in-time komputasi yang secara tepat memenuhi persyaratan beban kerja Anda. Karpenter secara otomatis menyediakan sumber daya komputasi baru berdasarkan persyaratan spesifik beban kerja cluster. Ini termasuk persyaratan komputasi, penyimpanan, akselerasi, dan penjadwalan. Amazon EKS mendukung cluster menggunakan Karpenter, meskipun Karpenter bekerja dengan kluster Kubernetes yang sesuai. Untuk informasi lebih lanjut, lihat dokumentasi [Karpenter](#).

Important

Karpenter adalah perangkat lunak open-source yang AWS pelanggan bertanggung jawab untuk menginstal, mengonfigurasi, dan mengelola di kluster Kubernetes mereka. AWS memberikan dukungan teknis saat Karpenter dijalankan tanpa dimodifikasi menggunakan versi yang kompatibel di kluster Amazon EKS. Sangat penting bahwa pelanggan menjaga ketersediaan dan keamanan pengontrol Karpenter serta prosedur pengujian yang tepat

saat memutakhirkannya atau cluster Kubernetes di mana ia berjalan, sama seperti perangkat lunak yang dikelola pelanggan lainnya. Tidak ada Perjanjian Tingkat Layanan (SLA) untuk Karpenter dan pelanggan bertanggung jawab untuk memastikan bahwa EC2 instans yang diluncurkan oleh Karpenter memenuhi persyaratan bisnis mereka.

Cluster Autoscaler

Kubernetes Autoscaler Klaster secara otomatis menyesuaikan jumlah simpul di klaster Anda ketika pod gagal atau dijadwal ulang ke simpul lain. Cluster Autoscaler menggunakan grup Auto Scaling. Untuk informasi selengkapnya, lihat [Cluster Autoscaler](#) di AWS.

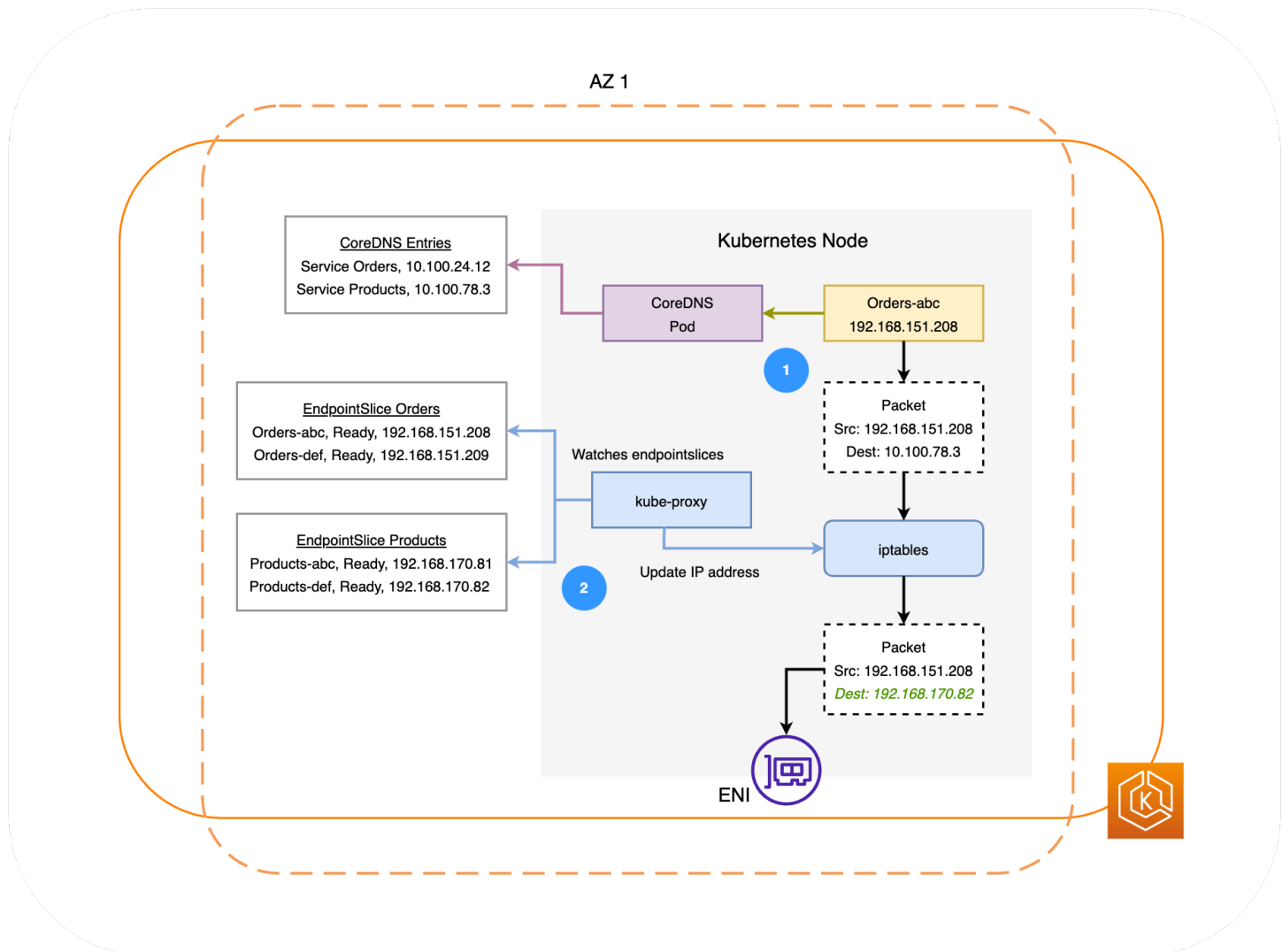
Pelajari tentang Pergeseran Zonal Pengontrol Pemulihan Aplikasi Amazon (ARC) di Amazon EKS

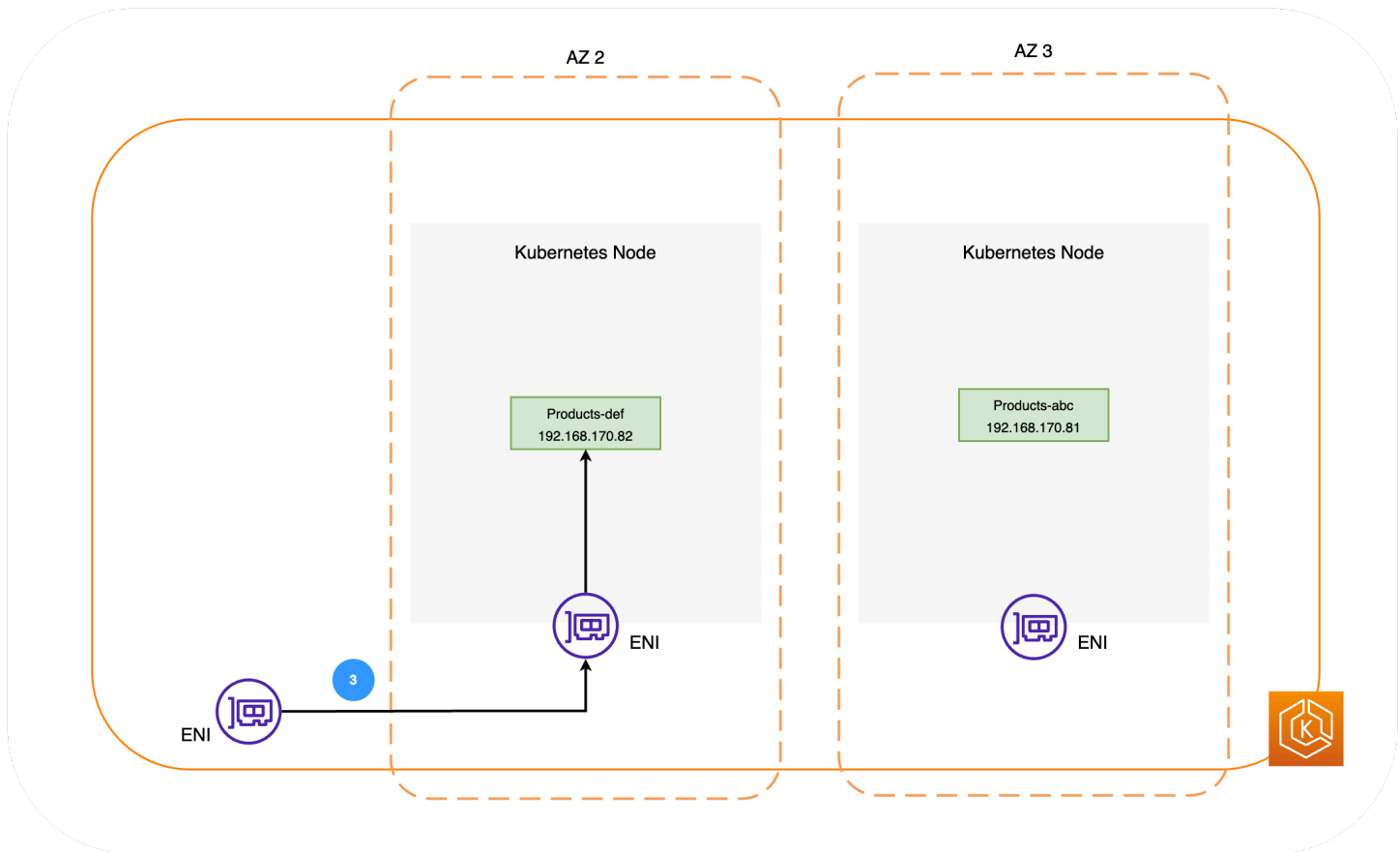
Kubernetes memiliki fitur asli yang memungkinkan Anda membuat aplikasi Anda lebih tahan terhadap peristiwa seperti penurunan kesehatan atau gangguan Availability Zone (AZ). [Saat menjalankan beban kerja di klaster Amazon EKS, Anda dapat lebih meningkatkan toleransi kesalahan lingkungan aplikasi dan pemulihan aplikasi menggunakan pergeseran zona Amazon Application Recovery Controller \(ARC\) atau pergeseran otomatis zona.](#) Pergeseran zona ARC dirancang untuk menjadi tindakan sementara yang memungkinkan Anda memindahkan lalu lintas untuk sumber daya dari AZ yang rusak hingga pergeseran zona berakhir atau Anda membatalkannya. Anda dapat memperpanjang pergeseran zona jika perlu.

Anda dapat memulai pergeseran zona untuk cluster EKS, atau Anda dapat mengizinkan melakukannya AWS untuk Anda dengan mengaktifkan pergeseran otomatis zona. Pergeseran ini memperbarui alur lalu lintas east-to-west jaringan di klaster Anda untuk hanya mempertimbangkan titik akhir jaringan untuk Pod yang berjalan di node pekerja dalam keadaan sehat AZs. Selain itu, ALB atau NLB apa pun yang menangani lalu lintas masuk untuk aplikasi di kluster EKS Anda akan secara otomatis merutekan lalu lintas ke target yang sehat. AZs Bagi pelanggan yang mencari tujuan ketersediaan tertinggi, jika AZ menjadi terganggu, penting untuk dapat mengarahkan semua lalu lintas dari AZ yang rusak hingga pulih. Untuk ini, Anda juga dapat [mengaktifkan ALB atau NLB dengan ARC](#) zonal shift.

Memahami Arus Lalu Lintas Jaringan Timur-Barat Antar Pod

Diagram berikut menggambarkan dua contoh beban kerja, Pesanan, dan Produk. Tujuan dari contoh ini adalah untuk menunjukkan bagaimana beban kerja dan Pod dalam AZs komunikasi yang berbeda.





1. Agar Pesanan dapat berkomunikasi dengan Produk, itu harus terlebih dahulu menyelesaikan nama DNS dari layanan tujuan. Pesanan akan berkomunikasi dengan CoreDNS untuk mengambil alamat IP virtual (Cluster IP) untuk Layanan tersebut. Setelah Pesanan menyelesaikan nama layanan Produk, ia mengirimkan lalu lintas ke IP target tersebut.
2. Kube-proxy berjalan pada setiap node di cluster dan terus mengawasi untuk Services.
[EndpointSlices](#) Ketika Layanan dibuat, sebuah EndpointSlice dibuat dan dikelola di latar belakang oleh EndpointSlice pengontrol. Masing-masing EndpointSlice memiliki daftar atau tabel titik akhir yang berisi subset alamat Pod bersama dengan node yang mereka jalankan. Kube-proxy mengatur aturan routing untuk masing-masing titik akhir Pod yang digunakan pada node. `iptables` Kube-proxy juga bertanggung jawab untuk bentuk dasar load balancing dengan mengarahkan lalu lintas yang ditujukan ke IP Cluster layanan untuk dikirim ke alamat IP Pod secara langsung. Kube-proxy melakukan ini dengan menulis ulang IP tujuan pada koneksi keluar.
3. Paket jaringan kemudian dikirim ke Products Pod di AZ 2 melalui node masing-masing (seperti yang digambarkan dalam diagram di atas). ENIs

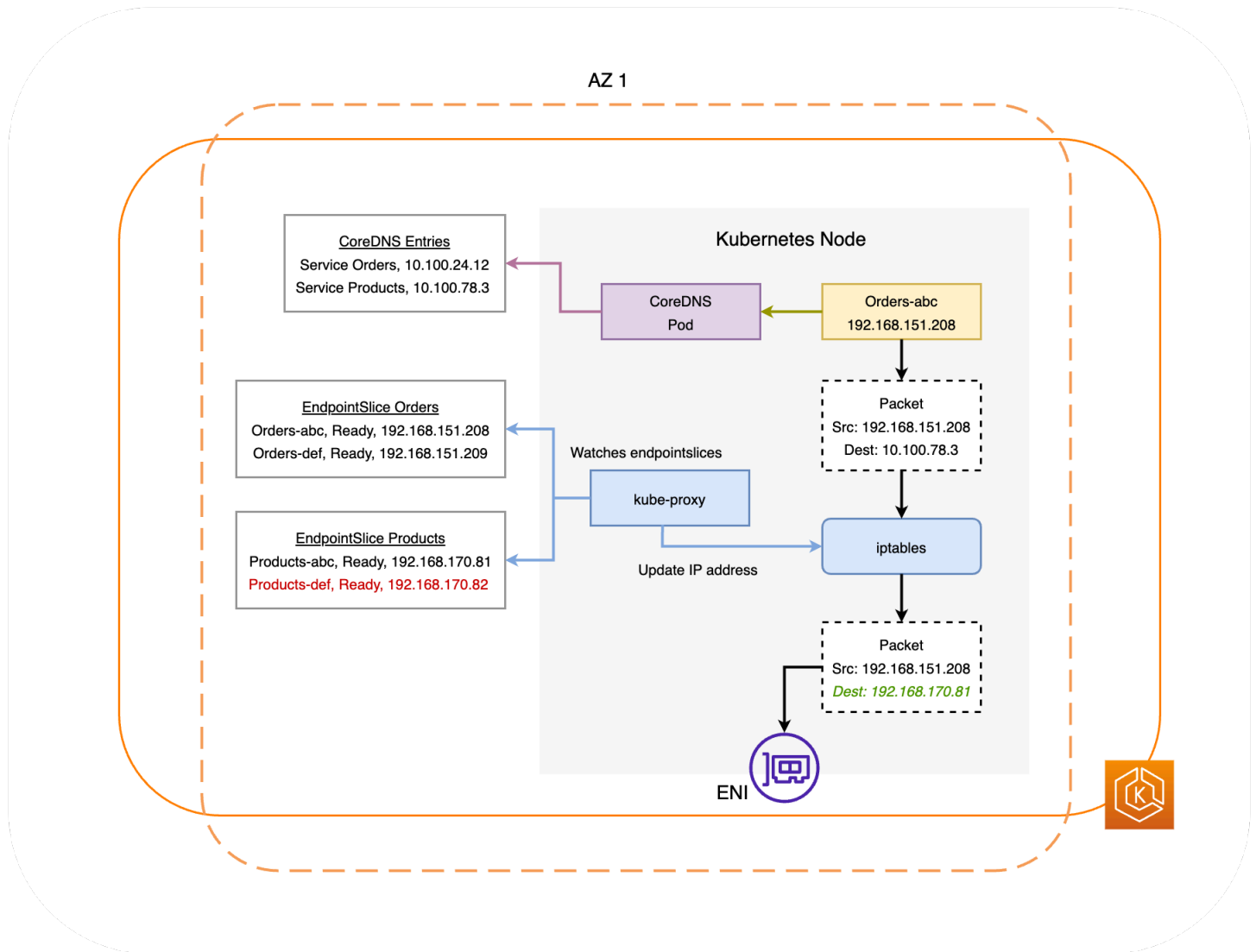
Memahami ARC Zonal Shift di Amazon EKS

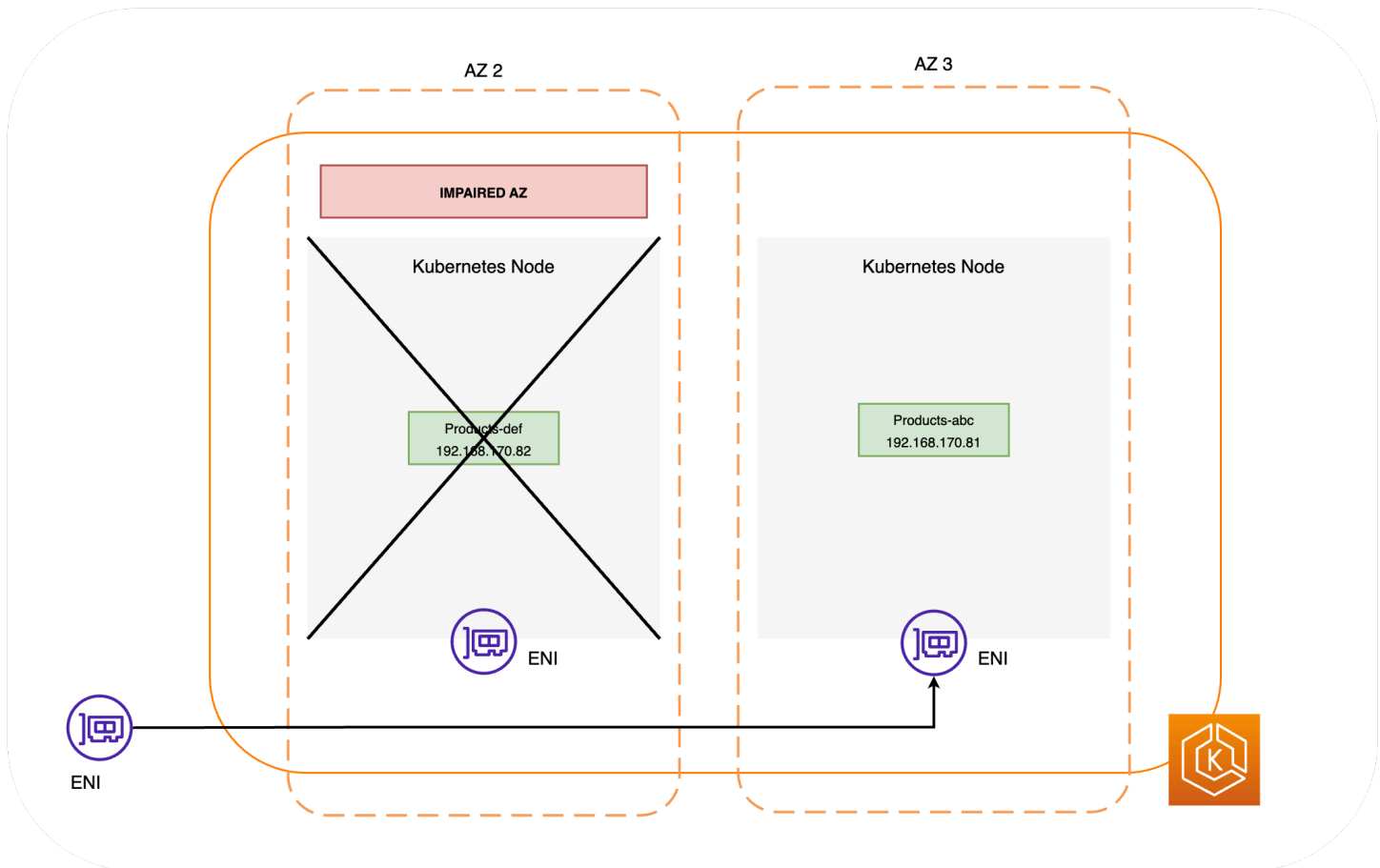
Jika ada gangguan AZ di lingkungan Anda, Anda dapat memulai pergeseran zona untuk lingkungan cluster EKS Anda. Atau, Anda dapat mengizinkan AWS untuk mengelola ini untuk Anda dengan zonal autoshift. Dengan pergeseran otomatis zona, AWS akan memantau kesehatan AZ secara keseluruhan dan merespons potensi gangguan AZ dengan secara otomatis mengalihkan lalu lintas dari AZ yang terganggu di lingkungan kluster Anda.

Setelah kluster Amazon EKS Anda mengaktifkan pergeseran zona dengan ARC, Anda dapat memicu pergeseran zona atau mengaktifkan pergeseran otomatis zona menggunakan Konsol ARC, AWS CLI, atau pergeseran zona dan pergeseran otomatis zona. APIs Selama pergeseran zona EKS, berikut ini akan secara otomatis terjadi:

- Semua node di AZ yang terkena dampak akan ditutup. Ini akan mencegah Kubernetes Scheduler menjadwalkan Pod baru ke node di AZ yang tidak sehat.
- Jika Anda menggunakan [Grup Node Terkelola](#), [penyeimbangan kembali Zona Ketersediaan](#) akan ditangguhkan, dan Grup Auto Scaling (ASG) Anda akan diperbarui untuk memastikan bahwa node Pesawat Data EKS baru hanya diluncurkan dalam kondisi sehat. AZs
- Node di AZ yang tidak sehat tidak akan dihentikan dan Pod tidak akan diusir dari node ini. Ini untuk memastikan bahwa ketika pergeseran zona berakhir atau dibatalkan, lalu lintas Anda dapat dikembalikan dengan aman ke AZ yang masih memiliki kapasitas penuh
- EndpointSlice Pengontrol akan menemukan semua titik akhir Pod di AZ yang rusak dan menghapusnya dari yang relevan EndpointSlices. Ini akan memastikan bahwa hanya titik akhir Pod yang sehat yang AZs ditargetkan untuk menerima lalu lintas jaringan. Ketika pergeseran zona dibatalkan atau kedaluwarsa, EndpointSlice pengontrol akan memperbarui EndpointSlices untuk menyertakan titik akhir di AZ yang dipulihkan.

Diagram di bawah ini menggambarkan aliran tingkat tinggi tentang bagaimana pergeseran zona EKS memastikan bahwa hanya titik akhir Pod yang sehat yang ditargetkan di lingkungan kluster Anda.





Persyaratan Pergeseran Zonal EKS

Agar pergeseran zona berhasil bekerja di EKS, Anda perlu mengatur lingkungan cluster Anda agar tahan terhadap gangguan AZ sebelumnya. Di bawah ini adalah daftar langkah-langkah yang harus Anda ikuti.

- Menyediakan node pekerja klaster Anda di beberapa AZs
- Menyediakan kapasitas komputasi yang cukup untuk menahan penghapusan AZ tunggal
- Pra-skala Pod Anda (termasuk CoreDNS) di setiap AZ
- Sebarkan beberapa replika Pod di semua AZs untuk memastikan bahwa pergeseran dari satu AZ akan membuat Anda memiliki kapasitas yang cukup
- Lokasi bersama Pod yang saling bergantung atau terkait di AZ yang sama
- Uji apakah lingkungan cluster Anda akan berfungsi seperti yang diharapkan dengan satu AZ lebih sedikit dengan memulai pergeseran zona secara manual. Atau, Anda dapat mengaktifkan pergeseran otomatis zonal dan membalas pada latihan autoshift. Ini tidak diperlukan agar pergeseran zona berfungsi di EKS tetapi sangat disarankan.

Menyediakan Node Pekerja EKS Anda di Beberapa AZs

AWS Wilayah memiliki beberapa lokasi terpisah dengan pusat data fisik yang dikenal sebagai Availability Zones (AZs). AZs dirancang untuk secara fisik terisolasi satu sama lain untuk menghindari dampak simultan yang dapat mempengaruhi seluruh Wilayah. Saat menyediakan kluster EKS, Anda harus menerapkan node pekerja Anda AZs di beberapa wilayah. Ini akan membuat lingkungan cluster Anda lebih tahan terhadap kerusakan AZ tunggal, dan memungkinkan Anda untuk mempertahankan ketersediaan tinggi (HA) aplikasi Anda yang berjalan di yang lain. AZs Saat Anda memulai pergeseran zona menjauh dari AZ yang terkena dampak, jaringan in-cluster lingkungan EKS Anda akan diperbarui secara otomatis untuk hanya menggunakan yang sehat AZs, sambil mempertahankan postur yang sangat tersedia untuk cluster Anda.

Memastikan bahwa Anda memiliki pengaturan Multi-AZ untuk lingkungan EKS Anda akan meningkatkan keandalan keseluruhan sistem Anda. Namun, lingkungan multi-AZ dapat memainkan peran penting dalam cara data aplikasi ditransfer dan diproses, yang pada gilirannya akan berdampak pada biaya jaringan lingkungan Anda. Secara khusus, lalu lintas lintas zona keluar yang sering (lalu lintas didistribusikan antara AZs) dapat berdampak besar pada biaya terkait jaringan Anda. Anda dapat menerapkan berbagai strategi untuk mengontrol jumlah lalu lintas lintas zona antar Pod di kluster EKS Anda dan menurunkan biaya terkait. Silakan lihat [panduan praktik terbaik ini](#) untuk detail lebih lanjut tentang cara mengoptimalkan biaya jaringan saat menjalankan lingkungan EKS yang sangat tersedia.

Diagram di bawah ini menggambarkan lingkungan EKS yang sangat tersedia dengan 3 sehat AZs.

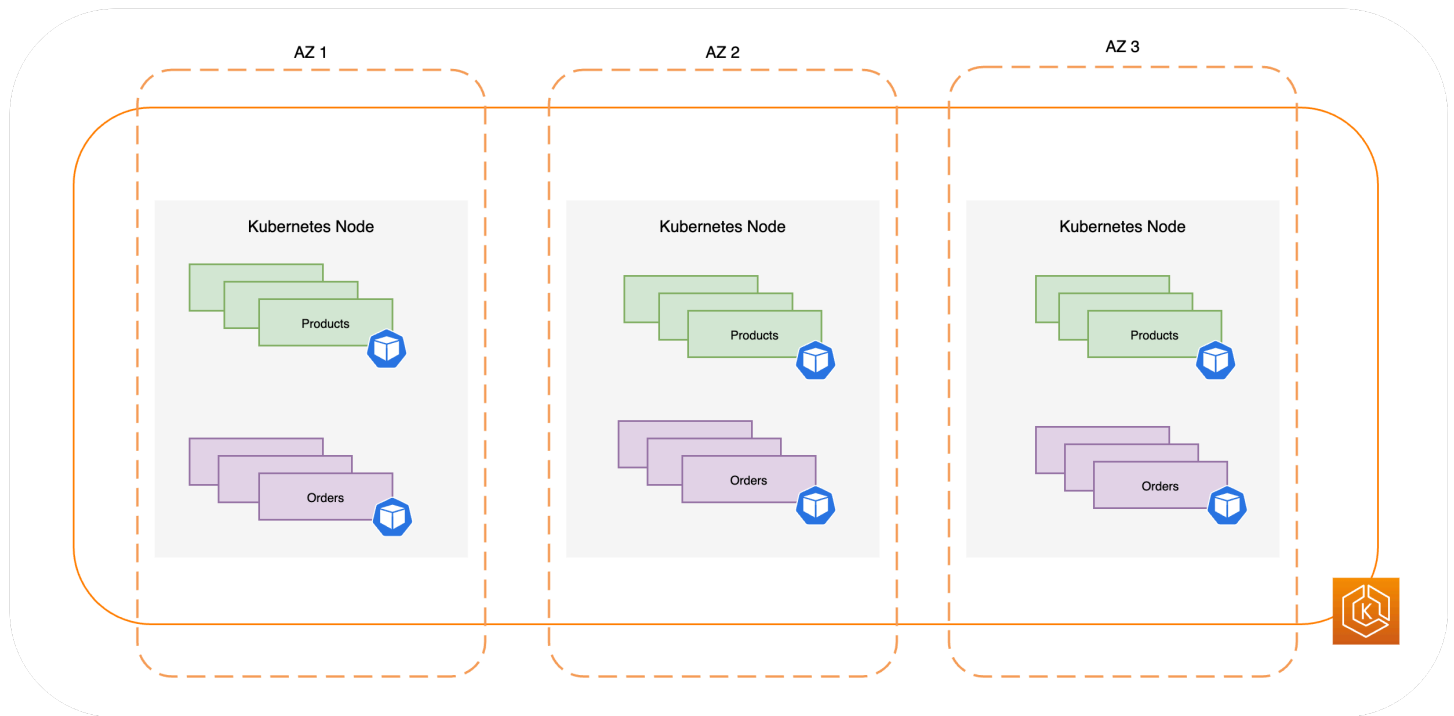
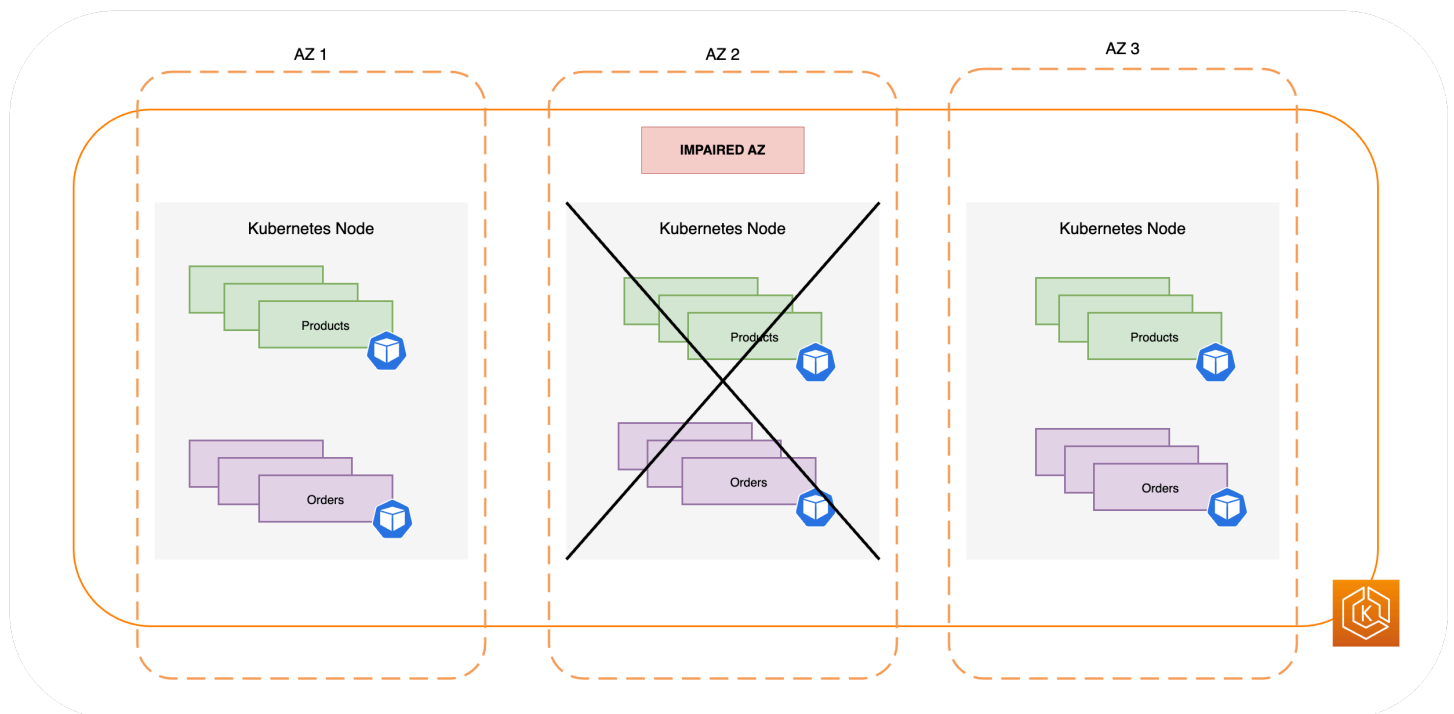


Diagram di bawah ini menggambarkan bagaimana lingkungan EKS dengan 3 AZs tahan terhadap gangguan AZ dan tetap sangat tersedia karena 2 lainnya sehat. AZs



Menyediakan Kapasitas Komputasi yang Cukup untuk Menahan Penghapusan AZ Tunggal

Untuk mengoptimalkan pemanfaatan sumber daya dan biaya untuk infrastruktur komputasi Anda di EKS Data Plane, ini adalah praktik terbaik untuk menyelaraskan kapasitas komputasi dengan persyaratan beban kerja Anda. Namun, jika semua node pekerja Anda berada pada kapasitas penuh, maka ini membuat Anda bergantung pada penambahan node pekerja baru ke Pesawat Data EKS sebelum Pod baru dapat dijadwalkan. Saat menjalankan beban kerja kritis, umumnya selalu merupakan praktik yang baik untuk dijalankan dengan kapasitas berlebihan secara online untuk menangani kemungkinan seperti peningkatan beban yang tiba-tiba, masalah kesehatan simpul, dll. Jika Anda berencana untuk menggunakan pergeseran zona, Anda berencana untuk menghapus seluruh kapasitas AZ sehingga Anda perlu menyesuaikan kapasitas komputasi redundan Anda sehingga cukup untuk menangani beban bahkan dengan AZ offline.

Saat menskalakan komputasi Anda, proses menambahkan node baru ke Pesawat Data EKS akan memakan waktu yang dapat berimplikasi pada kinerja real-time dan ketersediaan aplikasi Anda, terutama jika terjadi gangguan zona. Lingkungan EKS Anda harus tangguh untuk menyerap beban kehilangan AZ untuk menghindari pengalaman yang terdegradasi bagi pengguna akhir atau klien Anda. Ini berarti meminimalkan atau menghilangkan jeda antara waktu di mana Pod baru dibutuhkan dan kapan sebenarnya dijadwalkan pada node pekerja.

Selain itu, jika terjadi gangguan zona, Anda harus mengurangi risiko kendala kapasitas komputasi potensial yang akan mencegah node yang baru diperlukan ditambahkan ke Pesawat Data EKS Anda dalam kondisi sehat. AZs

Untuk mencapai hal ini, Anda harus menyediakan kapasitas komputasi yang berlebihan di beberapa node pekerja di masing-masing node AZs sehingga Kubernetes Scheduler memiliki kapasitas yang sudah ada sebelumnya yang tersedia untuk penempatan Pod baru, terutama ketika Anda memiliki satu AZ yang lebih sedikit di lingkungan Anda.

Jalankan & Sebarkan Beberapa Replika Pod Di Seluruh AZs

Kubernetes memungkinkan Anda untuk melakukan pra-skala beban kerja Anda dengan menjalankan beberapa instance (replika Pod) dari satu aplikasi. Menjalankan beberapa replika Pod untuk sebuah aplikasi menghilangkan satu titik kegagalan dan meningkatkan kinerjanya secara keseluruhan dengan mengurangi ketegangan sumber daya pada satu replika. Namun, untuk memiliki ketersediaan tinggi dan toleransi kesalahan yang lebih baik untuk aplikasi Anda, Anda harus menjalankan dan menyebarkan beberapa replika aplikasi di berbagai domain kegagalan (juga

disebut sebagai domain topologi) dalam kasus ini. AZs Dengan [kendala penyebaran topologi](#), Anda dapat mengatur aplikasi Anda agar memiliki stabilitas statis yang sudah ada sebelumnya sehingga, dalam kasus gangguan AZ, Anda akan memiliki cukup replika dalam kondisi sehat AZs untuk segera menangani lonjakan atau lonjakan lalu lintas tambahan yang mungkin mereka alami.

Diagram di bawah ini menggambarkan lingkungan EKS dengan arus east-to-west lalu lintas ketika AZs semuanya sehat.

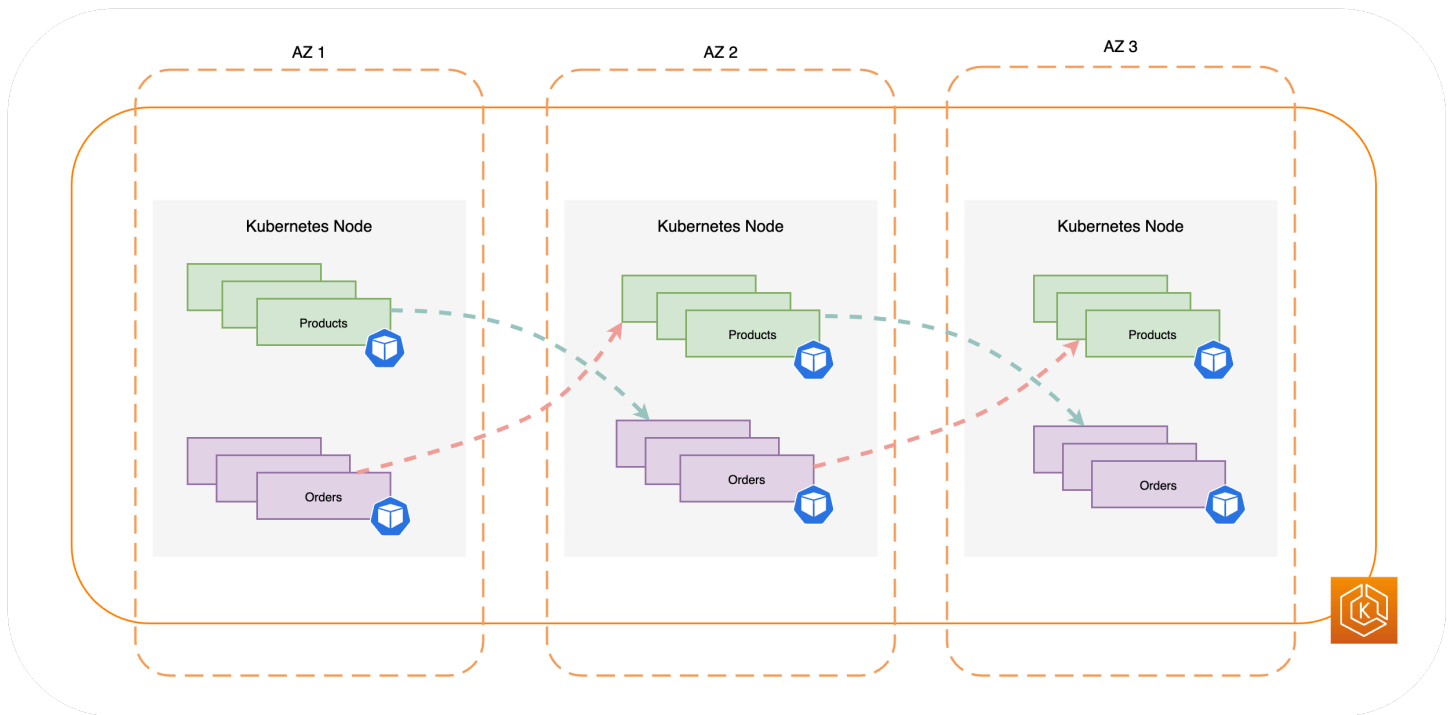
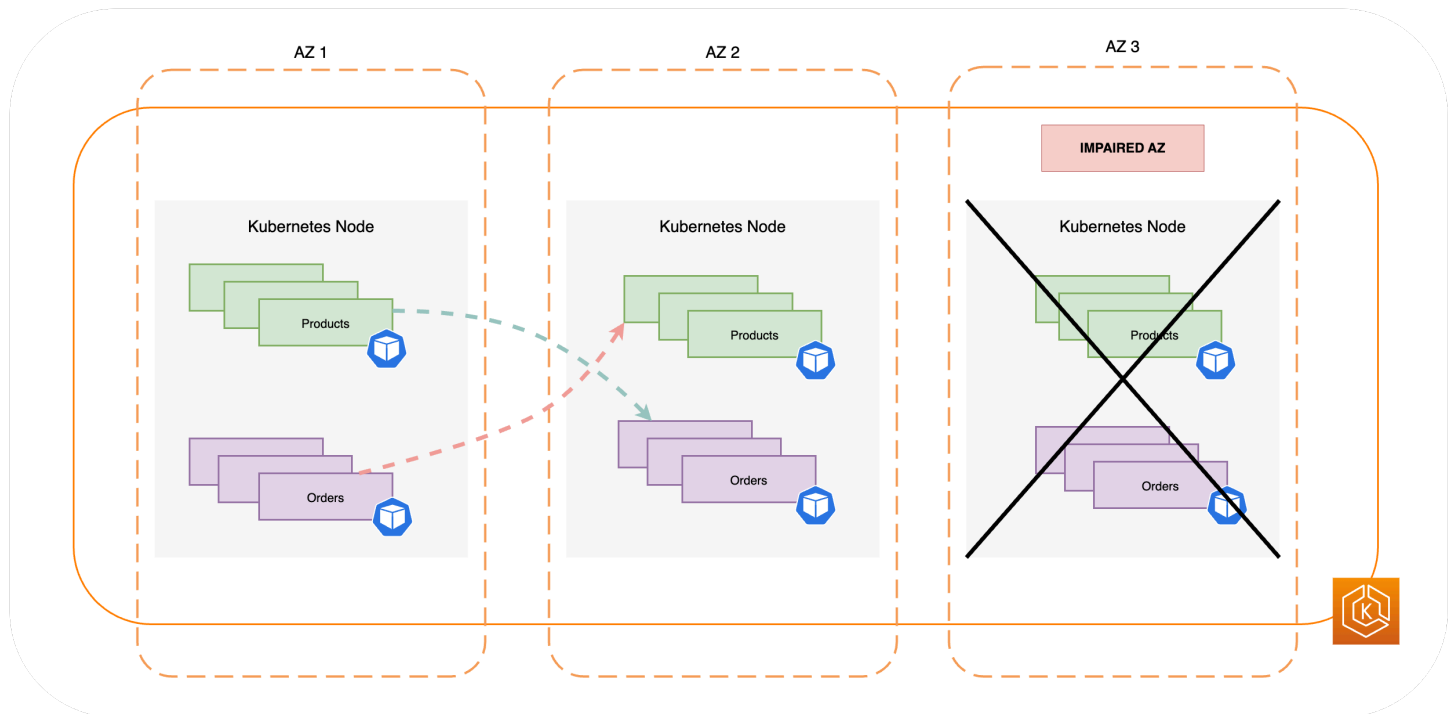


Diagram di bawah ini menggambarkan lingkungan EKS dengan arus east-to-west lalu lintas ketika satu AZ gagal, dan Anda memulai pergeseran zona.



Cuplikan kode di bawah ini adalah contoh cara mengatur beban kerja Anda dengan fitur Kubernetes ini.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: orders
spec:
  replicas: 9
  selector:
    matchLabels:
      app: orders
  template:
    metadata:
      labels:
        app: orders
        tier: backend
    spec:
      topologySpreadConstraints:
        - maxSkew: 1
          topologyKey: "topology.kubernetes.io/zone"
          whenUnsatisfiable: ScheduleAnyway
          labelSelector:
            matchLabels:
```

```
app: orders
```

Yang paling penting, Anda harus menjalankan beberapa replika perangkat lunak server DNS Anda (CoreDNS/Kube-DNS) dan menerapkan batasan penyebaran topologi serupa jika belum dikonfigurasi secara default. Ini akan membantu memastikan bahwa Anda memiliki cukup Pod DNS dalam kondisi sehat AZs untuk terus menangani permintaan penemuan layanan untuk Pod komunikasi lainnya di kluster jika ada satu gangguan AZ. Add-on [CoreDNS EKS](#) memiliki pengaturan default untuk Pod CoreDNS untuk tersebar di Availability Zone kluster Anda jika ada node dalam beberapa yang tersedia. AZs Anda juga dapat mengganti pengaturan default ini dengan konfigurasi kustom Anda sendiri.

Saat menginstal [CoreDNS dengan](#) Helm, Anda dapat memperbarui `replicaCount` file in [values.yaml](#) untuk memastikan bahwa Anda memiliki jumlah replika yang cukup di setiap AZ. Selain itu, untuk memastikan bahwa replika ini tersebar AZs di berbagai lingkungan kluster Anda, Anda harus memperbarui `topologySpreadConstraints` properti dalam file `values.yaml` yang sama. Cuplikan kode di bawah ini menunjukkan cara mengkonfigurasi CoreDNS untuk ini.

CoreDNS Helm nilai.yaml

```
replicaCount: 6
topologySpreadConstraints:
  - maxSkew: 1
    topologyKey: topology.kubernetes.io/zone
    whenUnsatisfiable: ScheduleAnyway
    labelSelector:
      matchLabels:
        k8s-app: kube-dns
```

Jika terjadi kerusakan AZ, Anda dapat menyerap peningkatan beban pada Pod CoreDNS dengan menggunakan sistem penskalaan otomatis untuk CoreDNS. Jumlah instans DNS yang Anda butuhkan akan bergantung pada jumlah beban kerja yang berjalan di kluster Anda. CoreDNS adalah CPU bound yang memungkinkannya untuk skala berdasarkan CPU menggunakan [Horizontal Pod Autoscaler](#) (HPA). Di bawah ini adalah contoh yang dapat Anda modifikasi sesuai dengan kebutuhan Anda.

```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: coredns
```



```
namespace: default
spec:
  maxReplicas: 20
  minReplicas: 2
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: coredns
  targetCPUUtilizationPercentage: 50
```

Atau, EKS dapat mengelola penskalaan otomatis Penerapan CoreDNS di CoreDNS versi add-on EKS. Autoscaler CoreDNS ini terus memantau status cluster, termasuk jumlah node dan core CPU. Berdasarkan informasi tersebut, pengontrol akan secara dinamis menyesuaikan jumlah replika penerapan CoreDNS di cluster EKS.

Untuk mengaktifkan [konfigurasi penskalaan otomatis di add-on CoreDNS EKS](#), Anda harus menambahkan pengaturan konfigurasi opsional berikut:

```
{
  "autoScaling": {
    "enabled": true
  }
}
```

Anda juga dapat menggunakan [NodeLocal DNS](#) atau [autoscaler proporsional cluster](#) untuk menskalakan CoreDNS. Anda dapat membaca lebih lanjut tentang [penskalaan CoreDNS](#) secara horizontal di sini.

Colocate Pod Interdependen di AZ yang Sama

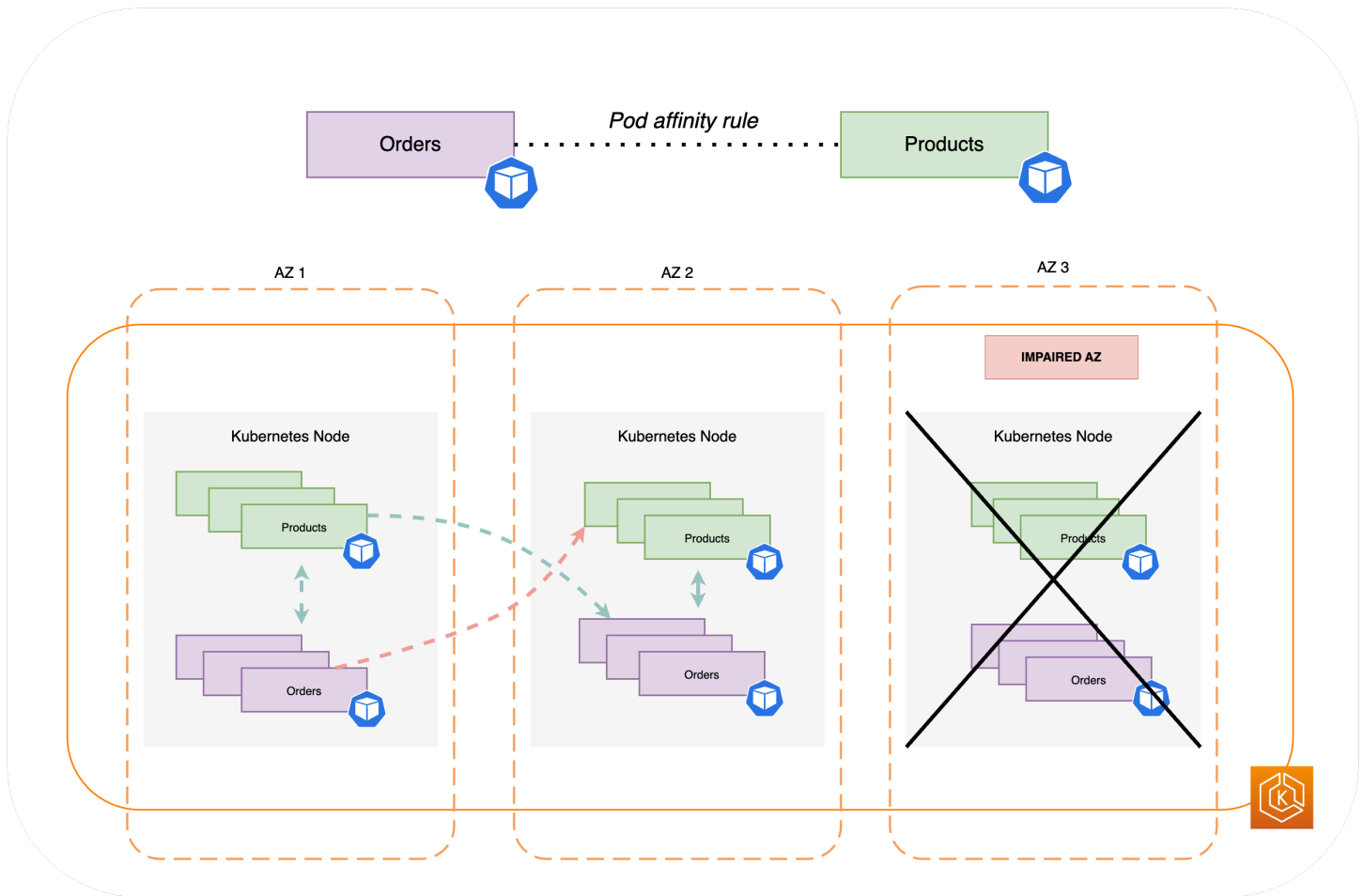
Dalam kebanyakan kasus, Anda mungkin menjalankan beban kerja yang berbeda yang harus berkomunikasi satu sama lain untuk keberhasilan pelaksanaan suatu end-to-end proses. Jika aplikasi yang berbeda tersebar di berbagai tempat AZs tetapi tidak ditempatkan di AZ yang sama, maka penurunan AZ tunggal dapat memengaruhi proses yang mendasarinya end-to-end. Misalnya, jika Aplikasi A memiliki beberapa replika di AZ 1 dan AZ 2, tetapi Aplikasi B memiliki semua replika di AZ 3, maka hilangnya AZ 3 akan memengaruhi end-to-end proses apa pun antara dua beban kerja ini (Aplikasi A dan B). Menggabungkan batasan penyebaran topologi dengan afinitas pod dapat meningkatkan ketahanan aplikasi Anda dengan menyebarkan Pod ke semua Pod AZs, serta mengonfigurasi hubungan antara Pod tertentu untuk memastikan bahwa mereka terkolokasi bersama.

Dengan [aturan afinitas pod](#), Anda dapat menentukan hubungan antar beban kerja untuk memengaruhi perilaku Penjadwal Kubernetes sehingga Colocates Pod pada node pekerja yang sama atau di AZ yang sama. Anda juga dapat mengonfigurasi seberapa ketat batasan penjadwalan ini seharusnya.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: products
  namespace: ecommerce
  labels:
    app.kubernetes.io/version: "0.1.6"

spec:
  serviceName: graphql-service-account
  affinity:
    podAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        - labelSelector:
            matchExpressions:
              - key: app
                operator: In
                values:
                  - orders
          topologyKey: "kubernetes.io/hostname"
```

Diagram di bawah ini menggambarkan pod yang telah ditempatkan bersama pada node yang sama menggunakan aturan afinitas pod.



Uji Bahwa Lingkungan Cluster Anda Dapat Menangani Hilangnya AZ

Setelah menyelesaikan persyaratan di atas, langkah penting berikutnya adalah menguji apakah Anda memiliki kapasitas komputasi dan beban kerja yang cukup untuk menangani hilangnya AZ. Anda dapat melakukan ini dengan secara manual memicu pergeseran zona di EKS. Atau, Anda dapat mengaktifkan zonal autoshift dan mengkonfigurasi praktik berjalan untuk menguji apakah aplikasi Anda berfungsi seperti yang diharapkan dengan satu AZ yang lebih sedikit di lingkungan cluster Anda.

Pertanyaan yang Sering Diajukan

Mengapa saya harus menggunakan fitur ini?

Dengan menggunakan ARC zonal shift atau zonal autoshift di kluster EKS Anda, Anda dapat lebih menjaga ketersediaan aplikasi Kubernetes dengan mengotomatiskan proses pemulihan cepat untuk mengalihkan lalu lintas jaringan in-cluster dari AZ yang rusak. Dengan ARC, Anda dapat menghindari

langkah-langkah panjang dan rumit yang sering menyebabkan periode pemulihan yang diperpanjang selama peristiwa AZ yang terganggu.

Bagaimana cara kerja fitur ini dengan AWS layanan lain?

EKS terintegrasi dengan ARC yang menyediakan antarmuka utama bagi Anda untuk menyelesaikan operasi pemulihan di. AWS Untuk memastikan bahwa lalu lintas in-cluster diarahkan secara tepat dari AZ yang rusak, modifikasi dilakukan pada daftar titik akhir jaringan untuk Pod yang berjalan di bidang data Kubernetes. Jika Anda menggunakan AWS Load Balancer untuk merutekan lalu lintas eksternal ke cluster, Anda sudah dapat mendaftarkan penyeimbang beban Anda dengan ARC dan memicu pergeseran zona pada mereka untuk mencegah lalu lintas mengalir ke zona terdegradasi. Fitur ini juga berinteraksi dengan Amazon EC2 Auto Scaling Groups (ASG) yang dibuat oleh EKS Managed Node Groups (MNG). Untuk mencegah AZ yang rusak digunakan untuk peluncuran Pod atau node Kubernetes baru, EKS menghapus AZ yang rusak dari ASG.

Apa perbedaan fitur ini dengan perlindungan Kubernetes default?

Fitur ini bekerja bersama-sama dengan beberapa perlindungan bawaan asli Kubernetes yang membantu pelanggan tetap tangguh. Anda dapat mengonfigurasi probe kesiapan dan keaktifan Pod yang menentukan kapan Pod harus mengambil lalu lintas. Ketika probe ini gagal, Kubernetes menghapus Pod ini sebagai target untuk Service dan lalu lintas tidak lagi dikirim ke Pod. Meskipun ini berguna, tidak sepele bagi pelanggan untuk mengonfigurasi pemeriksaan kesehatan ini sehingga dijamin gagal ketika suatu zona terdegradasi. Fitur ARC zonal shift memberi Anda jaring pengaman tambahan yang membantu mereka mengisolasi AZ yang terdegradasi sepenuhnya ketika perlindungan asli Kubernetes belum cukup. Ini juga memberi Anda cara mudah untuk menguji kesiapan operasional dan ketahanan arsitektur Anda.

Dapat AWS memicu pergeseran zona atas nama saya?

Ya, jika Anda menginginkan cara yang sepenuhnya otomatis menggunakan pergeseran zona ARC, Anda dapat mengaktifkan pergeseran otomatis zona ARC. Dengan zonal autoshift, Anda dapat mengandalkan AWS untuk memantau kesehatan kluster EKS Anda, dan untuk secara otomatis memicu pergeseran ketika gangguan AZ terdeteksi. AZs

Apa yang terjadi jika saya menggunakan fitur ini dan node pekerja serta beban kerja saya tidak diskalakan sebelumnya?

Jika Anda tidak melakukan pra-skala dan mengandalkan penyediaan node atau Pod tambahan selama pergeseran zona, maka Anda berisiko mengalami pemulihan yang tertunda. Proses penambahan node baru ke bidang data Kubernetes akan memakan waktu yang dapat berimplikasi

pada kinerja real-time dan ketersediaan aplikasi Anda, terutama jika terjadi kerusakan zona. Selain itu, jika terjadi gangguan zona, Anda mungkin menghadapi kendala kapasitas komputasi potensial yang akan mencegah node yang baru diperlukan ditambahkan ke yang sehat. AZs

Jika beban kerja Anda tidak diskalakan sebelumnya dan tersebar AZs di semua kluster Anda, gangguan zona dapat memengaruhi ketersediaan aplikasi yang hanya berjalan pada node pekerja di AZ yang terkena dampak. Untuk mengurangi risiko pemadaman ketersediaan lengkap untuk aplikasi Anda, EKS memiliki keamanan yang gagal untuk lalu lintas yang dikirim ke titik akhir Pod di zona yang terganggu jika beban kerja tersebut memiliki semua titik akhir di AZ yang tidak sehat. Namun, sangat disarankan agar Anda melakukan pra-skala dan menyebarkan aplikasi Anda AZs ke semua untuk menjaga ketersediaan jika terjadi masalah zona.

Apa yang terjadi jika saya menjalankan aplikasi stateful?

Jika Anda menjalankan aplikasi stateful, Anda perlu menilai toleransi kesalahannya tergantung pada kasus penggunaan dan arsitekturnya. Jika Anda memiliki arsitektur atau pola aktif/siaga, mungkin ada contoh di mana aktif berada di AZ yang rusak. Pada tingkat aplikasi, jika siaga tidak diaktifkan, Anda mungkin mengalami masalah dengan aplikasi Anda. Anda juga mungkin mengalami masalah ketika Pod Kubernetes baru diluncurkan dalam keadaan sehat AZs karena mereka tidak akan dapat melampirkan ke volume persisten yang dibatasi ke AZ yang rusak.

Apakah fitur ini berfungsi dengan Karpenter?

Dukungan Karpenter saat ini tidak tersedia dengan ARC zonal shift dan zonal autoshift di EKS. Jika AZ terganggu, Anda dapat menyesuaikan NodePool konfigurasi Karpenter yang relevan dengan menghapus AZ yang tidak sehat sehingga node pekerja baru hanya diluncurkan dalam kondisi sehat. AZs

Apakah fitur ini berfungsi dengan EKS Fargate?

Fitur ini tidak berfungsi dengan EKS Fargate. Secara default, ketika EKS Fargate mengenali peristiwa kesehatan zona, Pod akan lebih memilih untuk berjalan di acara lainnya. AZs

Apakah pesawat kontrol Kubernetes yang dikelola EKS akan terkena dampak?

Tidak, secara default Amazon EKS menjalankan dan menskalakan bidang kontrol Kubernetes di beberapa bidang AZs untuk memastikan ketersediaan yang tinggi. ARC zonal shift dan zonal autoshift hanya akan bekerja pada bidang data Kubernetes.

Apakah ada biaya yang terkait dengan fitur baru ini?

Anda dapat menggunakan ARC zonal shift dan zonal autoshift di kluster EKS Anda tanpa biaya tambahan. Namun, Anda akan terus membayar instans yang disediakan dan sangat disarankan agar Anda melakukan pra-skala pesawat data Kubernetes sebelum menggunakan fitur ini. Anda harus mempertimbangkan keseimbangan yang tepat antara biaya dan ketersediaan aplikasi.

Sumber Daya Tambahan

- [Cara kerja pergeseran zona](#)
- [Praktik terbaik untuk pergeseran zona di ARC](#)
- [Sumber daya dan skenario yang didukung untuk pergeseran zona dan pergeseran otomatis zona](#)
- [Mengoperasikan beban kerja tangguh di Amazon EKS](#)
- [Hilangkan lag penskalaan node Kubernetes dengan prioritas pod dan penyediaan berlebih](#)
- [Menskalakan Pod CoreDNS untuk lalu lintas DNS tinggi](#)

Aktifkan EKS Zonal Shift untuk menghindari gangguan Availability Zone

Amazon Application Recovery Controller (ARC) membantu Anda mengelola dan mengoordinasikan pemulihan untuk aplikasi Anda di seluruh Availability Zones (AZs) dan bekerja dengan banyak layanan, termasuk Amazon EKS. Dengan dukungan EKS untuk pergeseran zona ARC, Anda dapat mengalihkan lalu lintas jaringan in-cluster dari AZ yang rusak. Anda juga dapat mengizinkan AWS untuk memantau kesehatan Anda AZs dan mengalihkan lalu lintas jaringan untuk sementara dari AZ yang tidak sehat atas nama Anda.

Cara menggunakan EKS Zonal Shift:

1. Aktifkan kluster EKS Anda dengan Amazon Application Recovery Controller (ARC). Ini dilakukan di tingkat cluster menggunakan Amazon EKS Console, AWS CLI, CloudFormation, atau eksctl.
2. Setelah diaktifkan, Anda dapat mengelola pergeseran zona atau pergeseran otomatis zona menggunakan Konsol ARC, AWS CLI, atau Zonal Shift dan Zonal Autoshift. APIs

Perhatikan bahwa setelah Anda mendaftarkan cluster EKS dengan ARC, Anda masih perlu mengkonfigurasi ARC. Misalnya, Anda dapat menggunakan konsol ARC untuk mengkonfigurasi Zonal Autoshift.

Untuk informasi lebih rinci tentang cara kerja EKS Zonal Shift, dan cara mendesain beban kerja Anda untuk menangani zona ketersediaan yang terganggu, lihat. [the section called “Pelajari tentang Zonal Shift”](#)

Pertimbangan

- Mode Otomatis EKS tidak mendukung Amazon Application Recovery Controller, Zonal Shift, dan Zonal Autoshift.
- Kami merekomendasikan menunggu setidaknya 60 detik antara operasi pergeseran zona untuk memastikan pemrosesan yang tepat dari setiap permintaan.

Saat mencoba melakukan pergeseran zona secara berurutan (dalam waktu 60 detik satu sama lain), layanan Amazon EKS mungkin tidak memproses semua permintaan shift dengan benar. Hal ini disebabkan mekanisme polling saat ini yang memperbarui status zona kluster. Jika Anda perlu melakukan beberapa pergeseran zona, pastikan ada waktu yang cukup antara operasi bagi sistem untuk memproses setiap perubahan.

Apa itu Pengontrol Pemulihan Aplikasi Amazon?

Amazon Application Recovery Controller (ARC) membantu Anda mempersiapkan dan menyelesaikan pemulihan yang lebih cepat untuk aplikasi yang berjalan AWS. Pergeseran zona memungkinkan Anda pulih dengan cepat dari gangguan Availability Zone (AZ), dengan memindahkan lalu lintas sementara untuk sumber daya yang didukung dari AZ, menjadi sehat AZs di AWS Wilayah.

[Pelajari selengkapnya tentang Amazon Application Recovery Controller \(ARC\)](#)

Apa itu pergeseran zona?

Zonal shift adalah kemampuan di ARC yang memungkinkan Anda memindahkan lalu lintas untuk sumber daya seperti cluster EKS atau Elastic Load Balancer jauh dari Availability Zone di AWS Region untuk mengurangi masalah dengan cepat dan memulihkan aplikasi Anda dengan cepat. Anda mungkin memilih untuk mengalihkan lalu lintas, misalnya, karena penerapan yang buruk menyebabkan masalah latensi, atau karena Availability Zone terganggu. Pergeseran zona tidak memerlukan langkah konfigurasi lanjutan.

[Pelajari selengkapnya tentang ARC Zonal Shift](#)

Apa itu zonal autoshift?

Zonal autoshift adalah kemampuan di ARC yang dapat Anda aktifkan untuk mengotorisasi AWS untuk mengalihkan lalu lintas dari AZ untuk sumber daya yang didukung, atas nama Anda, menjadi sehat AZs di Wilayah. AWS AWS memulai pergeseran otomatis ketika telemetri internal menunjukkan bahwa ada gangguan pada satu AZ di Wilayah yang berpotensi berdampak pada pelanggan. Telemetri internal menggabungkan metrik dari berbagai sumber, termasuk AWS jaringan, dan layanan Amazon dan Elastic EC2 Load Balancing.

AWS mengakhiri pergeseran otomatis ketika indikator menunjukkan bahwa tidak ada lagi masalah atau masalah potensial.

[Pelajari selengkapnya tentang ARC Zonal Autoshift](#)

Apa yang dilakukan EKS selama autoshift?

EKS memperbarui konfigurasi jaringan untuk menghindari mengarahkan lalu lintas ke gangguan. AZs Selain itu, jika Anda menggunakan Grup Node Terkelola, EKS hanya akan meluncurkan node baru dalam kondisi sehat AZs selama pergeseran zona. Ketika shift berakhir atau dibatalkan, konfigurasi jaringan akan dipulihkan untuk menyertakan AZ yang sebelumnya terdeteksi sebagai tidak sehat.

[Pelajari lebih lanjut tentang EKS Zonal Shift.](#)

Daftarkan cluster EKS dengan Amazon Application Recovery Controller (ARC) (AWS konsol)

1. Temukan nama dan wilayah cluster EKS yang ingin Anda daftarkan dengan ARC.
2. Arahkan ke [konsol EKS](#) di wilayah tersebut, dan pilih klaster Anda.
3. Pada halaman Info cluster, pilih tab Ikhtisar.
4. Di bawah judul Zonal Shift, pilih tombol Kelola.
5. Pilih aktifkan atau nonaktifkan untuk EKS Zonal Shift.

Sekarang cluster EKS Anda terdaftar dengan ARC.

Jika Anda AWS ingin mendeteksi dan menghindari zona ketersediaan yang terganggu, Anda perlu mengkonfigurasi ARC Zonal Autoshift. Misalnya, Anda dapat melakukan ini di konsol ARC.

Langkah Berikutnya

- Pelajari cara [mengaktifkan pergeseran otomatis zona](#).
- Pelajari cara [memulai pergeseran zona](#) secara manual.

Pelajari cara kerja kontrol akses di Amazon EKS

Pelajari cara mengelola akses ke kluster Amazon EKS Anda. Menggunakan Amazon EKS membutuhkan pengetahuan tentang bagaimana Kubernetes dan AWS Identity and Access Management (AWS IAM) menangani kontrol akses.

Bagian ini meliputi:

[the section called “Akses API Kubernetes”](#)— Pelajari cara mengaktifkan aplikasi atau pengguna untuk melakukan autentikasi ke Kubernetes API. Anda dapat menggunakan entri akses, aws-auth ConfigMap, atau penyedia OIDC eksternal.

[the section called “Akses sumber daya kluster”](#)— Pelajari cara mengonfigurasi AWS Management Console untuk berkomunikasi dengan kluster Amazon EKS Anda. Gunakan konsol untuk melihat sumber daya Kubernetes di kluster, seperti namespace, node, dan Pod.

[the section called “Akses kluster dengan kubectl”](#)— Pelajari cara mengonfigurasi kubectl untuk berkomunikasi dengan cluster Amazon EKS Anda. Gunakan AWS CLI untuk membuat file kubeconfig.

[the section called “Akses beban kerja ke AWS”](#)— Pelajari cara mengaitkan akun layanan Kubernetes dengan AWS Peran IAM. Anda dapat menggunakan Pod Identity atau IAM Roles for Service Accounts (IRSA).

Tugas Umum

- Beri pengembang akses ke API Kubernetes. Lihat sumber daya Kubernetes di file. AWS Management Console
 - Solusi: [Gunakan entri akses](#) untuk mengaitkan izin RBAC Kubernetes dengan Pengguna atau Peran IAM. AWS
- Konfigurasi kubectl untuk berbicara dengan kluster Amazon EKS menggunakan Credentials. AWS
 - Solusi: Gunakan AWS CLI untuk [membuat file kubeconfig](#).
- Gunakan penyedia identitas eksternal, seperti Ping Identity, untuk mengautentikasi pengguna ke Kubernetes API.
 - Solusi: [Tautkan penyedia OIDC eksternal](#).
- Berikan beban kerja pada kluster Kubernetes Anda kemampuan untuk menelepon. AWS APIs

- Solusi: [Gunakan Pod Identity](#) untuk mengaitkan Peran AWS IAM ke Akun Layanan Kubernetes.

Latar Belakang

- [Pelajari cara kerja Akun Layanan Kubernetes.](#)
- [Tinjau Model Kubernetes Role Based Access Control \(RBAC\)](#)
- Untuk informasi selengkapnya tentang mengelola akses ke AWS sumber daya, lihat [Panduan Pengguna AWS IAM](#). Atau, ikuti [pelatihan pengantar gratis tentang penggunaan AWS IAM](#).

Pertimbangan untuk Mode Otomatis EKS

Mode Otomatis EKS terintegrasi dengan entri akses EKS Pod Identity dan EKS EKS.

- Mode Otomatis EKS menggunakan entri akses untuk memberikan izin pada bidang kontrol EKS Kubernetes. Misalnya, kebijakan akses mengaktifkan Mode Otomatis EKS untuk membaca informasi tentang titik akhir dan layanan jaringan.
 - Anda tidak dapat menonaktifkan entri akses pada kluster Mode Otomatis EKS.
 - Anda dapat secara opsional mengaktifkan. `aws-auth ConfigMap`
 - Entri akses untuk Mode Otomatis EKS dikonfigurasi secara otomatis. Anda dapat melihat entri akses ini, tetapi Anda tidak dapat memodifikasinya.
 - Jika Anda menggunakan NodeClass untuk membuat Peran IAM Node kustom, Anda perlu membuat entri akses untuk peran tersebut menggunakan kebijakan EKSAuto NodePolicy akses Amazon.
- Jika Anda ingin memberikan izin beban kerja untuk AWS layanan, gunakan EKS Pod Identity.
 - Anda tidak perlu menginstal agen Pod Identity pada kluster Mode Otomatis EKS.

Berikan akses kepada pengguna dan peran IAM ke Kubernetes APIs

Cluster Anda memiliki endpoint API Kubernetes. Kubectl menggunakan API ini. Anda dapat mengautentikasi ke API ini menggunakan dua jenis identitas:

- Prinsip AWS Identity and Access Management (IAM) and Access Management (peran atau pengguna) - Jenis ini memerlukan otentikasi ke IAM. Pengguna dapat masuk AWS sebagai

pengguna [IAM](#) atau dengan [identitas federasi](#) dengan menggunakan kredensial yang disediakan melalui sumber identitas. Pengguna hanya dapat masuk dengan identitas federasi jika administrator Anda sebelumnya menyiapkan federasi identitas menggunakan peran IAM. Ketika pengguna mengakses AWS dengan menggunakan federasi, mereka secara tidak langsung [mengambil peran](#). Saat pengguna menggunakan jenis identitas ini, Anda:

- Dapat menetapkan mereka izin Kubernetes sehingga mereka dapat bekerja dengan objek Kubernetes di kluster Anda. Untuk informasi selengkapnya tentang cara menetapkan izin ke prinsipal IAM Anda sehingga mereka dapat mengakses objek Kubernetes di kluster Anda, lihat [the section called “Berikan izin”](#)
- Dapat menetapkan mereka izin IAM sehingga mereka dapat bekerja dengan kluster Amazon EKS Anda dan sumber dayanya menggunakan Amazon EKS API, AWS CLI,,, atau. AWS CloudFormation AWS Management Consoleeksctl Untuk informasi selengkapnya, lihat [Tindakan yang ditentukan oleh Amazon Elastic Kubernetes Service di Referensi Otorisasi Layanan](#).
- Node bergabung dengan cluster Anda dengan mengasumsikan peran IAM. Kemampuan untuk mengakses kluster Anda menggunakan prinsipal IAM disediakan oleh [AWS IAM Authenticator for Kubernetes](#), yang berjalan di bidang kontrol Amazon EKS.
- Pengguna di penyedia OpenID Connect (OIDC) Anda sendiri — Jenis ini memerlukan otentikasi ke penyedia [OIDC Anda](#). Untuk informasi selengkapnya tentang menyiapkan penyedia OIDC Anda sendiri dengan kluster Amazon EKS Anda, lihat [the section called “Tautkan penyedia OIDC”](#) Saat pengguna menggunakan jenis identitas ini, Anda:
 - Dapat menetapkan mereka izin Kubernetes sehingga mereka dapat bekerja dengan objek Kubernetes di kluster Anda.
 - Tidak dapat menetapkan mereka izin IAM sehingga mereka dapat bekerja dengan kluster Amazon EKS Anda dan sumber dayanya menggunakan Amazon EKS API, AWS CLI,,, atau. AWS CloudFormation AWS Management Consoleeksctl

Anda dapat menggunakan kedua jenis identitas dengan cluster Anda. Metode otentikasi IAM tidak dapat dinonaktifkan. Metode otentikasi OIDC adalah opsional.

Kaitkan Identitas IAM dengan Izin Kubernetes

[AWS IAM Authenticator untuk Kubernetes](#) diinstal pada control plane cluster Anda. Ini memungkinkan prinsipal (peran [dan pengguna](#))[AWS Identity and Access Management](#) (IAM) yang Anda izinkan untuk mengakses sumber daya Kubernetes di kluster Anda. Anda dapat mengizinkan prinsipal IAM untuk mengakses objek Kubernetes di kluster Anda menggunakan salah satu metode berikut:

- Membuat entri akses — Jika klaster Anda berada pada atau lebih lambat dari versi platform yang tercantum di bagian [Prasyarat](#) untuk versi Kubernetes klaster Anda, kami sarankan Anda menggunakan opsi ini.

Gunakan entri akses untuk mengelola izin Kubernetes dari prinsipal IAM dari luar klaster. Anda dapat menambahkan dan mengelola akses ke cluster dengan menggunakan EKS API, AWS Command Line Interface AWS SDKs, AWS CloudFormation, dan AWS Management Console. Ini berarti Anda dapat mengelola pengguna dengan alat yang sama dengan yang Anda buat dengan cluster.

Untuk memulai, ikuti [Ubah mode autentikasi untuk menggunakan entri akses, lalu Memigrasi entri ConfigMap aws-auth yang ada untuk](#) mengakses entri.

- Menambahkan entri ke **aws-auth ConfigMap** — Jika versi platform cluster Anda lebih awal dari versi yang tercantum di bagian [Prasyarat](#), maka Anda harus menggunakan opsi ini. Jika versi platform klaster Anda pada atau lebih lambat dari versi platform yang tercantum di bagian [Prasyarat](#) untuk versi Kubernetes klaster Anda, dan Anda telah menambahkan entri ke ConfigMap, maka sebaiknya Anda memigrasikan entri tersebut untuk mengakses entri. ConfigMapNamun, Anda tidak dapat memigrasikan entri yang ditambahkan Amazon EKS, seperti entri untuk peran IAM yang digunakan dengan grup node terkelola atau profil Fargate. Untuk informasi selengkapnya, lihat [the section called “Akses API Kubernetes”](#).
- Jika Anda harus menggunakan aws-auth ConfigMap opsi, Anda dapat menambahkan entri ke ConfigMap menggunakan `eksctl create iamidentitymapping` perintah. Untuk informasi selengkapnya, lihat [Mengelola pengguna dan peran IAM](#) dalam eksctl dokumentasi.

Atur Mode Otentikasi Cluster

Setiap cluster memiliki mode otentikasi. Mode otentikasi menentukan metode mana yang dapat Anda gunakan untuk mengizinkan prinsipal IAM mengakses objek Kubernetes di klaster Anda. Ada tiga mode otentikasi.

Important

Setelah metode entri akses diaktifkan, itu tidak dapat dinonaktifkan.

Jika ConfigMap metode ini tidak diaktifkan selama pembuatan cluster, itu tidak dapat diaktifkan nanti. Semua cluster yang dibuat sebelum pengenalan entri akses memiliki ConfigMap metode yang diaktifkan.

Jika Anda menggunakan node hybrid dengan cluster Anda, Anda harus menggunakan mode otentikasi API atau API_AND_CONFIG_MAP cluster.

Bagian **aws-auth ConfigMap** dalam cluster

Ini adalah mode otentikasi asli untuk kluster Amazon EKS. Prinsipal IAM yang menciptakan cluster adalah pengguna awal yang dapat mengakses cluster dengan menggunakan `kubectl`. Pengguna awal harus menambahkan pengguna lain ke daftar di **aws-auth ConfigMap** dan menetapkan izin yang memengaruhi pengguna lain dalam cluster. Pengguna lain ini tidak dapat mengelola atau menghapus pengguna awal, karena tidak ada entri di **ConfigMap** to manage.

Baik entri **ConfigMap** dan akses

Dengan mode otentikasi ini, Anda dapat menggunakan kedua metode untuk menambahkan prinsip IAM ke cluster. Perhatikan bahwa setiap metode menyimpan entri terpisah; misalnya, jika Anda menambahkan entri akses dari AWS CLI, tidak **aws-auth ConfigMap** diperbarui.

Akses entri saja

Dengan mode otentikasi ini, Anda dapat menggunakan EKS API, AWS Command Line Interface, AWS SDKs, AWS CloudFormation, dan AWS Management Console untuk mengelola akses ke cluster untuk prinsipal IAM.

Setiap entri akses memiliki tipe dan Anda dapat menggunakan kombinasi cakupan akses untuk membatasi prinsipal ke namespace tertentu dan kebijakan akses untuk menetapkan kebijakan izin yang dapat digunakan kembali yang telah dikonfigurasi sebelumnya. Atau, Anda dapat menggunakan tipe **STANDARD** dan grup RBAC Kubernetes untuk menetapkan izin khusus.

Mode autentikasi	Metode
ConfigMap hanya (CONFIG_MAP)	aws-auth ConfigMap
EKS API dan ConfigMap (API_AND_CONFIG_MAP)	mengakses entri di EKS API, Antarmuka Baris AWS Perintah AWS SDKs, AWS CloudFormation, dan AWS Management Console aws-auth ConfigMap

Mode autentikasi	Metode
EKS API saja (API)	mengakses entri di EKS API, Antarmuka Baris AWS Perintah, AWS SDKs AWS CloudFormation, dan AWS Management Console

 Note

Mode Otomatis Amazon EKS memerlukan entri Akses.

Berikan akses kepada pengguna IAM ke Kubernetes dengan entri akses EKS

Apa itu entri akses EKS?

Entri akses EKS adalah cara terbaik untuk memberikan pengguna akses ke Kubernetes API. Misalnya, Anda dapat menggunakan entri akses untuk memberi pengembang akses untuk menggunakan kubectl.

Pada dasarnya, entri akses EKS mengaitkan satu set izin Kubernetes dengan identitas IAM, seperti peran IAM. Misalnya, pengembang dapat mengambil peran IAM dan menggunakannya untuk mengautentikasi ke Kluster EKS.

Anda dapat melampirkan izin Kubernetes untuk mengakses entri dengan dua cara:

- Gunakan kebijakan akses. Kebijakan akses adalah templat izin Kubernetes yang telah ditentukan sebelumnya yang dikelola oleh AWS. Untuk informasi selengkapnya, lihat [the section called “Tinjau kebijakan akses”](#).
- Referensi grup Kubernetes. Jika Anda mengaitkan Identitas IAM dengan grup Kubernetes, Anda dapat membuat sumber daya Kubernetes yang memberikan izin grup. Untuk informasi selengkapnya, lihat [Menggunakan Otorisasi RBAC](#) dalam dokumentasi Kubernetes.

Keuntungan

Manajemen akses klaster Amazon EKS memungkinkan Anda mengontrol autentikasi dan otorisasi untuk klaster Kubernetes Anda secara langsung melalui Amazon EKS. APIs Fitur ini

menyederhanakan manajemen akses dengan menghilangkan kebutuhan untuk beralih antara AWS dan Kubernetes APIs saat mengelola izin pengguna. Dengan menggunakan entri akses dan kebijakan akses, Anda dapat menentukan izin terperinci untuk prinsipal AWS IAM, termasuk kemampuan untuk memodifikasi atau mencabut izin cluster-admin dari pembuat kluster.

Fitur ini terintegrasi dengan alat infrastruktur sebagai kode (IaC) seperti AWS CloudFormation, Terraform, dan AWS CDK, memungkinkan Anda menentukan konfigurasi akses selama pembuatan cluster. Jika terjadi kesalahan konfigurasi, Anda dapat memulihkan akses kluster melalui Amazon EKS API tanpa memerlukan akses API Kubernetes langsung. Pendekatan terpusat ini mengurangi overhead operasional dan meningkatkan keamanan dengan memanfaatkan kemampuan AWS IAM yang ada seperti pencatatan CloudTrail audit dan otentikasi multi-faktor.

Mulai

1. Tentukan kebijakan Identitas dan Akses IAM yang ingin Anda gunakan.
 - [the section called “Tinjau kebijakan akses”](#)
2. Aktifkan Entri Akses EKS di kluster Anda. Konfirmasikan bahwa Anda memiliki versi platform yang didukung.
 - [the section called “Mode autentikasi”](#)
3. Buat entri akses yang mengaitkan Identitas IAM dengan izin Kubernetes.
 - [the section called “Buat entri akses”](#)
4. Otentikasi ke cluster menggunakan identitas IAM.
 - [the section called “Mengatur AWS CLI”](#)
 - [the section called “Siapkan kubectl dan eksctl”](#)

Konfigurasi akses kluster lama

Saat Anda mengaktifkan entri akses EKS pada kluster yang dibuat sebelum fitur ini diperkenalkan (cluster dengan versi platform awal lebih awal dari yang ditentukan dalam Persyaratan Versi Platform), EKS secara otomatis membuat entri akses yang mencerminkan izin yang sudah ada sebelumnya. Entri akses ini menunjukkan:

- Identitas IAM yang awalnya menciptakan cluster
- Izin administratif yang diberikan kepada identitas tersebut selama pembuatan kluster

 Note

Sebelumnya, akses administratif ini diberikan secara otomatis dan tidak dapat dimodifikasi. Dengan entri akses EKS diaktifkan, Anda sekarang dapat melihat dan menghapus konfigurasi akses lama ini.

Mengaitkan kebijakan akses dengan entri akses

Anda dapat menetapkan satu atau beberapa kebijakan akses untuk mengakses jenis entri. STANDARD Amazon EKS secara otomatis memberikan jenis entri akses lainnya izin yang diperlukan agar berfungsi dengan baik di klaster Anda. Kebijakan akses Amazon EKS mencakup izin Kubernetes, bukan izin IAM. Sebelum mengaitkan kebijakan akses ke entri akses, pastikan Anda sudah terbiasa dengan izin Kubernetes yang disertakan dalam setiap kebijakan akses. Untuk informasi selengkapnya, lihat [the section called “Tinjau kebijakan akses”](#). Jika tidak ada kebijakan akses yang memenuhi persyaratan Anda, maka jangan kaitkan kebijakan akses ke entri akses. Sebagai gantinya, tentukan satu atau beberapa nama grup untuk entri akses dan buat serta kelola objek kontrol akses berbasis peran Kubernetes. Untuk informasi selengkapnya, lihat [the section called “Buat entri akses”](#).

- Entri akses yang ada. Untuk membuatnya, lihat [the section called “Buat entri akses”](#).
- Peran AWS Identity and Access Management atau pengguna dengan izin berikut: `ListAccessEntries`, `DescribeAccessEntry`, `UpdateAccessEntry`, `ListAccessPolicies`, `AssociateAccessPolicy`, dan `DisassociateAccessPolicy`. Untuk informasi selengkapnya, lihat [Tindakan yang ditentukan oleh Amazon Elastic Kubernetes Service di Referensi Otorisasi Layanan](#).

Sebelum mengaitkan kebijakan akses dengan entri akses, pertimbangkan persyaratan berikut:

- Anda dapat mengaitkan beberapa kebijakan akses ke setiap entri akses, tetapi Anda hanya dapat mengaitkan setiap kebijakan ke entri akses satu kali. Jika Anda mengaitkan beberapa kebijakan akses, prinsipal IAM entri akses memiliki semua izin yang disertakan dalam semua kebijakan akses terkait.
- Anda dapat membuat cakupan kebijakan akses ke semua sumber daya di klaster atau dengan menentukan nama satu atau beberapa ruang nama Kubernetes. Anda dapat menggunakan karakter wildcard untuk nama namespace. Misalnya, jika Anda ingin membuat cakupan kebijakan akses ke semua ruang nama yang dimulaidev-, Anda dapat menentukan dev-* sebagai nama

namespace. Pastikan bahwa namespace ada di cluster Anda dan ejaan Anda cocok dengan nama namespace yang sebenarnya di cluster. Amazon EKS tidak mengonfirmasi ejaan atau keberadaan ruang nama di cluster Anda.

- Anda dapat mengubah cakupan akses untuk kebijakan akses setelah Anda mengaitkannya ke entri akses. Jika Anda telah mencakup kebijakan akses ke ruang nama Kubernetes, Anda dapat menambahkan dan menghapus namespace untuk asosiasi tersebut, jika diperlukan.
- Jika Anda mengaitkan kebijakan akses ke entri akses yang juga memiliki nama grup yang ditentukan, maka prinsipal IAM memiliki semua izin di semua kebijakan akses terkait. Ia juga memiliki semua izin di Kubernetes Role atau ClusterRole objek apa pun yang ditentukan dalam Kubernetes Role dan RoleBinding objek apa pun yang menentukan nama grup.
- Jika Anda menjalankan `kubectl auth can-i --list` perintah, Anda tidak akan melihat izin Kubernetes yang ditetapkan oleh kebijakan akses yang terkait dengan entri akses untuk prinsipal IAM yang Anda gunakan saat menjalankan perintah. Perintah hanya menampilkan izin Kubernetes jika Anda telah memberikannya di Kubernetes Role atau ClusterRole objek yang telah Anda ikat ke nama grup atau nama pengguna yang Anda tentukan untuk entri akses.
- Jika Anda meniru pengguna atau grup Kubernetes saat berinteraksi dengan objek Kubernetes di kluster Anda, seperti menggunakan `kubectl` perintah dengan `--as username` atau `--as-group group-name`, Anda memaksa penggunaan otorisasi Kubernetes RBAC. Akibatnya, prinsipal IAM tidak memiliki izin yang ditetapkan oleh kebijakan akses apa pun yang terkait dengan entri akses. Satu-satunya izin Kubernetes yang dimiliki pengguna atau grup yang ditiru oleh kepala sekolah IAM adalah izin Kubernetes yang Anda berikan kepada mereka di Kubernetes Role atau objek yang Anda ikat ke nama grup atau nama pengguna. ClusterRole Agar prinsipal IAM Anda memiliki izin dalam kebijakan akses terkait, jangan menyamar sebagai pengguna atau grup Kubernetes. Prinsipal IAM juga akan tetap memiliki izin apa pun yang Anda berikan di Kubernetes Role atau ClusterRole objek yang telah Anda ikat ke nama grup atau nama pengguna yang Anda tentukan untuk entri akses. Untuk informasi selengkapnya, lihat [Peniruan identitas pengguna di dokumentasi](#) Kubernetes.

Anda dapat mengaitkan kebijakan akses ke entri akses menggunakan AWS Management Console atau AWS CLI.

AWS Management Console

1. Buka [konsol Amazon EKS](#).
2. Pilih nama kluster yang memiliki entri akses yang ingin Anda kaitkan dengan kebijakan akses.
3. Pilih tab Access.

4. Jika jenis entri akses adalah Standar, Anda dapat mengaitkan atau memisahkan kebijakan akses Amazon EKS. Jika jenis entri akses Anda adalah apa pun selain Standar, maka opsi ini tidak tersedia.
5. Pilih Kebijakan akses asosiasi.
6. Untuk nama Kebijakan, pilih kebijakan dengan izin yang Anda inginkan untuk memiliki kepala sekolah IAM. Untuk melihat izin yang disertakan dalam setiap kebijakan, lihat [the section called “Tinjau kebijakan akses”](#).
7. Untuk cakupan Access, pilih cakupan akses. Jika Anda memilih Cluster, izin dalam kebijakan akses diberikan kepada prinsipal IAM untuk sumber daya di semua ruang nama Kubernetes. Jika Anda memilih namespace Kubernetes, Anda dapat memilih Add new namespace. Di bidang Namespace yang muncul, Anda dapat memasukkan nama namespace Kubernetes di kluster Anda. Jika Anda ingin prinsipal IAM memiliki izin di beberapa ruang nama, maka Anda dapat memasukkan beberapa ruang nama.
8. Pilih Tambahkan kebijakan akses.

AWS CLI

1. Versi 2.12.3 atau yang lebih baru atau versi 1.27.160 atau yang lebih baru dari AWS Command Line Interface (AWS CLI) diinstal dan dikonfigurasi pada perangkat Anda atau AWS CloudShell. Untuk memeriksa versi Anda saat ini, gunakan `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Package manager seperti `yum` atau `apt-get`, atau Homebrew untuk macOS seringkali merupakan beberapa versi di belakang versi terbaru CLI. AWS Untuk menginstal versi terbaru, lihat [Menginstal](#) dan [Konfigurasi cepat dengan aws configure](#) di Panduan Pengguna Antarmuka Baris AWS Perintah. Versi AWS CLI yang diinstal AWS CloudShell mungkin juga beberapa versi di belakang versi terbaru. Untuk memperbaruinya, lihat [Menginstal AWS CLI ke direktori home Anda](#) di AWS CloudShell Panduan Pengguna.
2. Lihat kebijakan akses yang tersedia.

```
aws eks list-access-policies --output table
```

Contoh output adalah sebagai berikut.

```
-----
|                               ListAccessPolicies
```

```

+-----+
+
||                                     accessPolicies
||
||                                     ||
|+-----+
+-----+|
||                                     arn                                |
name                                     ||
|+-----+
+-----+|
|| {arn-aws}eks::aws:cluster-access-policy/AmazonEKSAAdminPolicy      |
AmazonEKSAAdminPolicy                ||
|| {arn-aws}eks::aws:cluster-access-policy/AmazonEKSClusterAdminPolicy |
AmazonEKSClusterAdminPolicy          ||
|| {arn-aws}eks::aws:cluster-access-policy/AmazonEKSEditPolicy         |
AmazonEKSEditPolicy                  ||
|| {arn-aws}eks::aws:cluster-access-policy/AmazonEKSVIEWPolicy         |
AmazonEKSVIEWPolicy                  ||
|+-----+
+-----+|

```

Untuk melihat izin yang disertakan dalam setiap kebijakan, lihat [the section called “Tinjau kebijakan akses”](#).

3. Lihat entri akses yang ada. Ganti *my-cluster* dengan nama kluster Anda.

```
aws eks list-access-entries --cluster-name my-cluster
```

Contoh output adalah sebagai berikut.

```
{
  "accessEntries": [
    "arn:aws:iam::111122223333:role/my-role",
    "arn:aws:iam::111122223333:user/my-user"
  ]
}
```

4. Kaitkan kebijakan akses ke entri akses. Contoh berikut mengaitkan kebijakan AmazonEKSVIEWPolicy akses ke entri akses. Setiap kali peran *my-role* IAM mencoba mengakses objek Kubernetes di kluster, Amazon EKS akan mengotorisasi peran tersebut untuk menggunakan izin dalam kebijakan untuk mengakses objek Kubernetes di ruang nama dan Kubernetes saja. *my-namespace1 my-namespace2* Ganti *my-cluster* dengan nama kluster

Anda, **111122223333** dengan ID AWS akun Anda, dan **my-role** dengan nama peran IAM yang Anda inginkan Amazon EKS untuk mengotorisasi akses ke objek cluster Kubernetes.

```
aws eks associate-access-policy --cluster-name my-cluster --principal-arn arn:aws:iam::111122223333:role/my-role \
    --access-scope type=namespace,namespaces=my-namespace1,my-namespace2 --policy-arn arn:aws:eks::aws:cluster-access-policy/AmazonEKSVViewPolicy
```

Jika Anda ingin kepala sekolah IAM memiliki izin di seluruh klaster, ganti dengan. `type=namespace,namespaces=`**my-namespace1,my-namespace2** `type=cluster` Jika Anda ingin mengaitkan beberapa kebijakan akses ke entri akses, jalankan perintah beberapa kali, masing-masing dengan kebijakan akses unik. Setiap kebijakan akses terkait memiliki ruang lingkupnya sendiri.

Note

Jika nanti Anda ingin mengubah cakupan kebijakan akses terkait, jalankan perintah sebelumnya lagi dengan cakupan baru. Misalnya, jika Anda ingin menghapus **my-namespace2**, Anda akan menjalankan perintah lagi `type=namespace,namespaces=`**my-namespace1** hanya dengan menggunakan. Jika Anda ingin mengubah cakupan dari namespace ke cluster, Anda akan menjalankan perintah lagi menggunakan `type=cluster`, menghapus `type=namespace,namespaces=`**my-namespace1,my-namespace2**.

5. Tentukan kebijakan akses mana yang terkait dengan entri akses.

```
aws eks list-associated-access-policies --cluster-name my-cluster --principal-arn arn:aws:iam::111122223333:role/my-role
```

Contoh output adalah sebagai berikut.

```
{
  "clusterName": "my-cluster",
  "principalArn": "arn:aws:iam::111122223333",
  "associatedAccessPolicies": [
    {
      "policyArn": "arn:aws:eks::aws:cluster-access-policy/AmazonEKSVViewPolicy",
      "accessScope": {
```

```

        "type": "cluster",
        "namespaces": []
    },
    "associatedAt": "2023-04-17T15:25:21.675000-04:00",
    "modifiedAt": "2023-04-17T15:25:21.675000-04:00"
},
{
    "policyArn": "arn:aws:eks::aws:cluster-access-policy/
AmazonEKSAAdminPolicy",
    "accessScope": {
        "type": "namespace",
        "namespaces": [
            "my-namespace1",
            "my-namespace2"
        ]
    },
    "associatedAt": "2023-04-17T15:02:06.511000-04:00",
    "modifiedAt": "2023-04-17T15:02:06.511000-04:00"
}
]
}

```

Pada contoh sebelumnya, prinsipal IAM untuk entri akses ini memiliki izin tampilan di semua ruang nama di kluster, dan izin administrator ke dua ruang nama Kubernetes.

- Memutuskan kebijakan akses dari entri akses. Dalam contoh ini, AmazonEKSAAdminPolicy kebijakan dilepaskan dari entri akses. Prinsipal IAM mempertahankan izin dalam kebijakan AmazonEKSVIEWPolicy akses untuk objek di *my-namespace1* dan *my-namespace2* ruang nama, karena kebijakan akses tersebut tidak terlepas dari entri akses.

```

aws eks disassociate-access-policy --cluster-name my-cluster --principal-arn arn:aws:
iam::111122223333:role/my-role \
    --policy-arn arn:aws:eks::aws:cluster-access-policy/AmazonEKSAAdminPolicy

```

Untuk mencantumkan kebijakan akses yang tersedia, lihat [the section called “Tinjau kebijakan akses”](#).

Migrasi entri yang ada untuk **aws-auth ConfigMap** mengakses entri

Jika Anda telah menambahkan entri ke **aws-auth ConfigMap** pada kluster Anda, kami sarankan Anda membuat entri akses untuk entri yang ada di **aws-auth ConfigMap**. Setelah membuat entri akses, Anda dapat menghapus entri dari entri Anda. **ConfigMap** Anda tidak dapat mengaitkan

[kebijakan akses](#) ke entri di `aws-auth ConfigMap` Jika Anda ingin mengaitkan kebijakan akses ke prinsipal IAM Anda, buat entri akses.

Important

- Saat cluster berada dalam mode `API_AND_CONFIGMAP` otentikasi dan ada pemetaan untuk peran IAM yang sama di entri akses `aws-auth ConfigMap` dan dalam entri akses, peran tersebut akan menggunakan pemetaan entri akses untuk otentikasi. Entri akses lebih diutamakan daripada `ConfigMap` entri untuk prinsipal IAM yang sama.
- Sebelum menghapus `aws-auth ConfigMap` entri yang ada yang dibuat oleh Amazon EKS untuk [grup node terkelola](#) atau profil [Fargate](#) ke kluster Anda, periksa kembali apakah entri akses yang benar untuk sumber daya spesifik tersebut ada di kluster Amazon EKS Anda. Jika Anda menghapus entri yang dibuat Amazon EKS `ConfigMap` tanpa memiliki entri akses yang setara, kluster Anda tidak akan berfungsi dengan baik.

Prasyarat

- Keakraban dengan entri akses dan kebijakan akses. Untuk informasi selengkapnya, lihat [the section called “Berikan izin”](#) dan [the section called “Kebijakan akses asosiasi”](#).
- Cluster yang ada dengan versi platform yang pada atau lebih lambat dari versi yang tercantum dalam Prasyarat topik. [the section called “Berikan izin”](#)
- Versi `0.210.0` atau yang lebih baru dari alat baris perintah yang diinstal pada perangkat Anda atau AWS CloudShell. Untuk menginstal atau memperbaruieksc1, lihat [Instalasi](#) dalam `eksctl` dokumentasi.
- Izin Kubernetes untuk memodifikasi di `aws-auth ConfigMap` namespace. `kube-system`
- Peran AWS Identity and Access Management atau pengguna dengan izin berikut: `CreateAccessEntry` dan `ListAccessEntries`. Untuk informasi selengkapnya, lihat [Tindakan yang ditentukan oleh Amazon Elastic Kubernetes Service di Referensi Otorisasi Layanan](#).

eksctl

1. Lihat entri yang ada di Anda `aws-auth ConfigMap`. Ganti *my-cluster* dengan nama kluster Anda.

```
eksctl get iamidentitymapping --cluster my-cluster
```

Contoh output adalah sebagai berikut.

ARN	USERNAME	GROUPS	ACCOUNT
arn:aws: iam::111122223333:role/EKS-my-cluster-Admins	Admins	system:masters	
arn:aws: iam::111122223333:role/EKS-my-cluster-my-namespace-Viewers	my-namespace-Viewers	Viewers	
arn:aws: iam::111122223333:role/EKS-my-cluster-self-managed-ng-1	system:node:{{EC2PrivateDNSName}}	system:bootstrappers,system:nodes	
arn:aws: iam::111122223333:user/my-user	my-user		
arn:aws: iam::111122223333:role/EKS-my-cluster-fargateprofile1	system:node:{{SessionName}}		
	system:bootstrappers,system:nodes,system:node-proxier		
arn:aws: iam::111122223333:role/EKS-my-cluster-managed-ng	system:node:{{EC2PrivateDNSName}}	system:bootstrappers,system:nodes	

2. [the section called “Buat entri akses”](#) untuk salah satu ConfigMap entri yang Anda buat kembali di output sebelumnya. Saat membuat entri akses, pastikan untuk menentukan nilai yang sama untuk ARN, USERNAME, dan GROUPS, dan ACCOUNT dikembalikan dalam output Anda. Dalam contoh keluaran, Anda akan membuat entri akses untuk semua entri kecuali dua entri terakhir, karena entri tersebut dibuat oleh Amazon EKS untuk profil Fargate dan grup node terkelola.
3. Hapus entri dari entri akses apa pun yang Anda ConfigMap buat. Jika Anda tidak menghapus entri dari ConfigMap, pengaturan untuk entri akses untuk ARN utama IAM akan menggantikan ConfigMap entri. Ganti **111122223333** dengan ID AWS akun Anda dan **EKS-my-cluster-my-namespace-Viewers** dengan nama peran dalam entri di akun Anda ConfigMap. Jika entri yang Anda hapus adalah untuk pengguna IAM, bukan peran IAM, ganti **role** dengan **user** dan **EKS-my-cluster-my-namespace-Viewers** dengan nama pengguna.

```
eksctl delete iamidentitymapping --arn arn:aws: iam::111122223333:role/EKS-my-cluster-my-namespace-Viewers --cluster my-cluster
```

Tinjau izin kebijakan akses

Kebijakan akses termasuk rules yang berisi Kubernetes verbs (izin) dan resources. Kebijakan akses tidak menyertakan izin atau sumber daya IAM. Mirip dengan Kubernetes Role dan

ClusterRole objek, kebijakan akses hanya menyertakan. `allow rules` Anda tidak dapat mengubah konten kebijakan akses. Anda tidak dapat membuat kebijakan akses sendiri. Jika izin dalam kebijakan akses tidak memenuhi kebutuhan Anda, buat objek Kubernetes RBAC dan tentukan nama grup untuk entri akses Anda. Untuk informasi selengkapnya, lihat [the section called “Buat entri akses”](#). Izin yang terdapat dalam kebijakan akses mirip dengan izin di peran klaster yang dihadapi pengguna Kubernetes. Untuk informasi selengkapnya, lihat [Peran yang dihadapi pengguna dalam dokumentasi](#) Kubernetes.

Pilih kebijakan akses apa pun untuk melihat isinya. Setiap baris dari setiap tabel di setiap kebijakan akses adalah aturan terpisah.

EKSAdminKebijakan Amazon

Kebijakan akses ini mencakup izin yang memberikan izin utama IAM untuk sumber daya. Ketika dikaitkan dengan entri akses, cakupan aksesnya biasanya satu atau beberapa ruang nama Kubernetes. Jika Anda ingin prinsipal IAM memiliki akses administrator ke semua sumber daya di klaster Anda, kaitkan kebijakan [the section called “Amazon EKSCluster AdminPolicy”](#) akses ke entri akses Anda.

ARN– `arn:aws:eks::aws:cluster-access-policy/AmazonEKSAdminPolicy`

Grup API Kubernetes	Sumber daya Kubernetes	Kata kerja Kubernetes (izin)
apps	daemonsets , deployments , deployments/rollback , deployments/scale , replicaset , replicaset/scale , statefulsets , statefulsets/scale	create, delete, deletecollection , patch, update
apps	controllerrevisions , daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , replicaset , replicaset/scale , replicaset	get, list, watch

Grup API Kubernetes	Sumber daya Kubernetes	Kata kerja Kubernetes (izin)
	ts/status , statefuls ets , statefulsets/ scale , statefulsets/ status	
authorization.k8s.io	localsubjectaccess reviews	create
autoscaling	horizontalpodautos calers	create, delete, deletecol lection , patch, update
autoscaling	horizontalpodautos calers , horizonta lpodautoscalers/st atus	get, list, watch
batch	cronjobs, jobs	create, delete, deletecol lection , patch, update
batch	cronjobs, cronjobs/ status , jobs, jobs/stat us	get, list, watch
discovery.k8s.io	endpointslices	get, list, watch
extensions	daemonsets , deploymen ts , deployments/ rollback , deploymen ts/scale , ingresses , networkpolicies , replicasets , replicase ts/scale , replicati oncontrollers/scale	create, delete, deletecol lection , patch, update

Grup API Kubernetes	Sumber daya Kubernetes	Kata kerja Kubernetes (izin)
extensions	daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , ingresses , ingresses/status , networkpolicies , replicaset , replicaset/scale , replicaset/status , replicationcontrollers/scale	get, list, watch
networking.k8s.io	ingresses , ingresses/status , networkpolicies	get, list, watch
networking.k8s.io	ingresses , networkpolicies	create, delete, deletecollection , patch, update
policy	poddisruptionbudgets	create, delete, deletecollection , patch, update
policy	poddisruptionbudgets , poddisruptionbudgets/status	get, list, watch
rbac.authorization.k8s.io	rolebindings , roles	create, delete, deletecollection , get, list, patch, update, watch

Grup API Kubernetes	Sumber daya Kubernetes	Kata kerja Kubernetes (izin)
	configmaps , endpoints , persistentvolumeclaims , persistentvolumeclaims/status , pods, replicationcontrollers , replicationcontrollers/scale , serviceaccounts , services, services/status	get,list, watch
	pods/attach , pods/exec , pods/portforward , pods/proxy , secrets, services/proxy	get, list, watch
	configmaps , events, persistentvolumeclaims , replicationcontrollers , replicationcontrollers/scale , secrets, serviceaccounts , services, services/proxy	create, delete, deletecollection , patch, update
	pods, pods/attach , pods/exec , pods/portforward , pods/proxy	create, delete, deletecollection , patch, update
	serviceaccounts	impersonate

Grup API Kubernetes	Sumber daya Kubernetes	Kata kerja Kubernetes (izin)
	bindings, events, limitranges , namespaces/status , pods/log, pods/status , replicationcontrollers/status , resourcequotas , resourcequotas/status	get, list, watch
	namespaces	get,list, watch

Amazon EKSCluster AdminPolicy

Kebijakan akses ini mencakup izin yang memberikan akses administrator utama IAM ke klaster. Ketika dikaitkan dengan entri akses, cakupan aksesnya biasanya adalah klaster, bukan namespace Kubernetes. Jika Anda ingin kepala sekolah IAM memiliki ruang lingkup administratif yang lebih terbatas, pertimbangkan untuk mengaitkan kebijakan [the section called “EKSClusterAdminPolicy”](#) akses ke entri akses Anda.

ARN– `arn:aws:eks::aws:cluster-access-policy/AmazonEKSClusterAdminPolicy`

Grup API Kubernetes	Kubernetes NonResource URLs	Sumber daya Kubernetes	Kata kerja Kubernetes (izin)
*		*	*
	*		*

Amazon EKSCluster AdminViewPolicy

Kebijakan akses ini mencakup izin yang memberikan akses utama IAM ke list/view semua sumber daya dalam klaster. Perhatikan bahwa ini termasuk Rahasia [Kubernetes](#).

ARN– `arn:aws:eks::aws:cluster-access-policy/AmazonEKSClusterAdminViewPolicy`

Grup API Kubernetes	Sumber daya Kubernetes	Kata kerja Kubernetes (izin)
*	*	get, list, watch

EKSEditKebijakan Amazon

Kebijakan akses ini mencakup izin yang memungkinkan prinsipal IAM untuk mengedit sebagian besar sumber daya Kubernetes.

ARN– `arn:aws:eks::aws:cluster-access-policy/AmazonEKSEditPolicy`

Grup API Kubernetes	Sumber daya Kubernetes	Kata kerja Kubernetes (izin)
apps	daemonsets , deployments , deployments/rollback , deployments/scale , replicaset , replicaset/scale , statefulsets , statefulsets/scale	create, delete, deletecollection , patch, update
apps	controllerrevisions , daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , replicaset , replicaset/scale , replicaset/status , statefulsets , statefulsets/scale , statefulsets/status	get, list, watch
autoscaling	horizontalpodautoscalers , horizontalpodautoscalers/status	get, list, watch

Grup API Kubernetes	Sumber daya Kubernetes	Kata kerja Kubernetes (izin)
autoscaling	horizontalpodautoscalers	create, delete, deletecollection , patch, update
batch	cronjobs, jobs	create, delete, deletecollection , patch, update
batch	cronjobs, cronjobs/status , jobs, jobs/status	get, list, watch
discovery.k8s.io	endpointslices	get, list, watch
extensions	daemonsets , deployments , deployments/rollback , deployments/scale , ingresses , networkpolicies , replicaset , replicaset/scale , replicationcontrollers/scale	create, delete, deletecollection , patch, update
extensions	daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , ingresses , ingresses/status , networkpolicies , replicaset , replicaset/scale , replicaset/status , replicationcontrollers/scale	get, list, watch
networking.k8s.io	ingresses , networkpolicies	create, delete, deletecollection , patch, update

Grup API Kubernetes	Sumber daya Kubernetes	Kata kerja Kubernetes (izin)
networking.k8s.io	ingresses , ingresses /status , networkpolicies	get, list, watch
policy	poddisruptionbudgets	create, delete, deletecollection , patch, update
policy	poddisruptionbudgets , poddisruptionbudgets/status	get, list, watch
	namespaces	get, list, watch
	pods/attach , pods/exec , pods/portforward , pods/proxy , secrets, services/proxy	get, list, watch
	serviceaccounts	impersonate
	pods, pods/attach , pods/exec , pods/portforward , pods/proxy	create, delete, deletecollection , patch, update
	configmaps , events, persistentvolumeclaims , replicationcontrollers , replicationcontrollers/scale , secrets, serviceaccounts , services, services/proxy	create, delete, deletecollection , patch, update

Grup API Kubernetes	Sumber daya Kubernetes	Kata kerja Kubernetes (izin)
	configmaps , endpoints , persistentvolumeclaims , persistentvolumeclaims/status , pods, replicationcontrollers , replicationcontrollers/scale , serviceaccounts , services, services/status	get, list, watch
	bindings, events, limitranges , namespaces/status , pods/log, pods/status , replicationcontrollers/status , resourcequotas , resourcequotas/status	get, list, watch

EKSViewKebijakan Amazon

Kebijakan akses ini mencakup izin yang memungkinkan prinsipal IAM untuk melihat sebagian besar sumber daya Kubernetes.

ARN– `arn:aws:eks::aws:cluster-access-policy/AmazonEKSViewPolicy`

Grup API Kubernetes	Sumber daya Kubernetes	Kata kerja Kubernetes (izin)
apps	controllerrevisions , daemonsets , daemonsets/status , deployments ts , deployments/scale , deployments/status , replicasets , replicase	get, list, watch

Grup API Kubernetes	Sumber daya Kubernetes	Kata kerja Kubernetes (izin)
	ts/scale , replicase ts/status , statefuls ets , statefulsets/ scale , statefulsets/ status	
autoscaling	horizontalpodautos calers , horizonta lpodautoscalers/st atus	get, list, watch
batch	cronjobs, cronjobs/ status , jobs, jobs/stat us	get, list, watch
discovery.k8s.io	endpointslices	get, list, watch
extensions	daemonsets , daemonset s/status , deploymen ts , deployments/scale , deployments/status , ingresses , ingresses /status , networkpo licies , replicaset s , replicaset s/scale , replicaset s/status , replicationcontrol lers/scale	get, list, watch
networking.k8s.io	ingresses , ingresses /status , networkpo licies	get, list, watch
policy	poddisruptionbudgets , poddisruptionbudge ts/status	get, list, watch

Grup API Kubernetes	Sumber daya Kubernetes	Kata kerja Kubernetes (izin)
	configmaps , endpoints , persistentvolumeclaims , persistentvolumeclaims/status , pods, replicationcontrollers , replicationcontrollers/scale , serviceaccounts , services, services/status	get, list, watch
	bindings, events, limitranges , namespaces/ status, pods/log, pods/ status , replicationcontrollers/ status , resourcequotas , resourcequotas/status	get, list, watch
	namespaces	get, list, watch

Amazon EKSAuto NodePolicy

ARN— `arn:aws:eks::aws:cluster-access-policy/AmazonEKSAutoNodePolicy`

Kebijakan ini mencakup izin berikut yang memungkinkan komponen Amazon EKS menyelesaikan tugas-tugas berikut:

- **kube-proxy**— Pantau titik akhir dan layanan jaringan, dan kelola acara terkait. Ini memungkinkan fungsionalitas proxy jaringan di seluruh cluster.
- **ipamd**— Kelola sumber daya jaringan AWS VPC dan antarmuka jaringan kontainer (CNI). Hal ini memungkinkan daemon manajemen alamat IP untuk menangani jaringan pod.
- **coredns**— Akses sumber daya penemuan layanan seperti titik akhir dan layanan. Ini memungkinkan resolusi DNS dalam cluster.

- `ebs-csi-driver`— Bekerja dengan sumber daya terkait penyimpanan untuk volume Amazon EBS. Hal ini memungkinkan penyediaan dinamis dan lampiran volume persisten.
- `neuron`— Pantau node dan pod untuk perangkat AWS Neuron. Hal ini memungkinkan pengelolaan AWS akselerator Inferentia dan Trainium.
- `node-monitoring-agent`— Akses diagnostik dan acara node. Hal ini memungkinkan pemantauan kesehatan cluster dan pengumpulan diagnostik.

Setiap komponen menggunakan akun layanan khusus dan dibatasi hanya untuk izin yang diperlukan untuk fungsi spesifiknya.

Jika Anda secara manual menentukan peran IAM Node dalam a NodeClass, Anda perlu membuat Entri Akses yang mengaitkan peran IAM Node baru dengan Kebijakan Akses ini.

Amazon EKSBLOCK StoragePolicy

Note

Kebijakan ini ditujukan hanya untuk peran AWS terkait layanan dan tidak dapat digunakan dengan peran yang dikelola pelanggan.

ARN— `arn:aws:eks::aws:cluster-access-policy/AmazonEKSBLOCKStoragePolicy`

Kebijakan ini mencakup izin yang memungkinkan Amazon EKS mengelola pemilihan pemimpin dan sumber daya koordinasi untuk operasi penyimpanan:

- `coordination.k8s.io`— Membuat dan mengelola objek sewa untuk pemilihan pemimpin. Hal ini memungkinkan komponen penyimpanan EKS untuk mengoordinasikan kegiatan mereka di seluruh cluster melalui mekanisme pemilihan pemimpin.

Kebijakan ini mencakup sumber daya sewa khusus yang digunakan oleh komponen penyimpanan EKS untuk mencegah akses yang bertentangan ke sumber daya koordinasi lain di cluster.

Amazon EKS secara otomatis membuat entri akses dengan kebijakan akses ini untuk peran IAM klaster saat Mode Otomatis diaktifkan, memastikan bahwa izin yang diperlukan tersedia agar kemampuan penyimpanan blok berfungsi dengan baik.

Amazon EKSLoad BalancingPolicy

ARN– `arn:aws:eks::aws:cluster-access-policy/AmazonEKSLoadBalancingPolicy`

Kebijakan ini mencakup izin yang memungkinkan Amazon EKS mengelola sumber daya pemilihan pemimpin untuk penyeimbangan beban:

- `coordination.k8s.io`— Membuat dan mengelola objek sewa untuk pemilihan pemimpin. Hal ini memungkinkan komponen penyeimbang beban EKS untuk mengoordinasikan aktivitas di beberapa replika dengan memilih pemimpin.

Kebijakan ini dicakup secara khusus untuk memuat penyeimbangan sumber daya sewa untuk memastikan koordinasi yang tepat sekaligus mencegah akses ke sumber daya sewa lainnya di cluster.

Amazon EKS secara otomatis membuat entri akses dengan kebijakan akses ini untuk peran IAM klaster saat Mode Otomatis diaktifkan, memastikan bahwa izin yang diperlukan tersedia agar kemampuan jaringan berfungsi dengan baik.

EKSNetworkingKebijakan Amazon

ARN– `arn:aws:eks::aws:cluster-access-policy/AmazonEKSNetworkingPolicy`

Kebijakan ini mencakup izin yang memungkinkan Amazon EKS mengelola sumber daya pemilihan pemimpin untuk jaringan:

- `coordination.k8s.io`— Membuat dan mengelola objek sewa untuk pemilihan pemimpin. Hal ini memungkinkan komponen jaringan EKS untuk mengoordinasikan aktivitas alokasi alamat IP dengan memilih pemimpin.

Kebijakan ini dicakup secara khusus untuk jaringan sumber daya sewa untuk memastikan koordinasi yang tepat sambil mencegah akses ke sumber daya sewa lainnya di cluster.

Amazon EKS secara otomatis membuat entri akses dengan kebijakan akses ini untuk peran IAM klaster saat Mode Otomatis diaktifkan, memastikan bahwa izin yang diperlukan tersedia agar kemampuan jaringan berfungsi dengan baik.

EKSComputeKebijakan Amazon

Note

Kebijakan ini ditujukan hanya untuk peran AWS terkait layanan dan tidak dapat digunakan dengan peran yang dikelola pelanggan.

ARN– `arn:aws:eks::aws:cluster-access-policy/AmazonEKSComputePolicy`

Kebijakan ini mencakup izin yang memungkinkan Amazon EKS mengelola sumber daya pemilihan pemimpin untuk operasi komputasi:

- `coordination.k8s.io`— Membuat dan mengelola objek sewa untuk pemilihan pemimpin. Hal ini memungkinkan komponen komputasi EKS untuk mengoordinasikan aktivitas penskalaan node dengan memilih pemimpin.

Kebijakan ini dicakup secara khusus untuk menghitung sumber daya sewa manajemen sambil memungkinkan akses baca dasar (`get,watch`) ke semua sumber daya sewa di cluster.

Amazon EKS secara otomatis membuat entri akses dengan kebijakan akses ini untuk peran IAM kluster saat Mode Otomatis diaktifkan, memastikan bahwa izin yang diperlukan tersedia agar kemampuan jaringan berfungsi dengan baik.

Amazon EKSBLOCK StorageClusterPolicy

Note

Kebijakan ini ditujukan hanya untuk peran AWS terkait layanan dan tidak dapat digunakan dengan peran yang dikelola pelanggan.

ARN– `arn:aws:eks::aws:cluster-access-policy/AmazonEKSBLOCKStorageClusterPolicy`

Kebijakan ini memberikan izin yang diperlukan untuk kemampuan penyimpanan blok Mode Otomatis Amazon EKS. Ini memungkinkan pengelolaan sumber daya penyimpanan blok yang efisien dalam kluster Amazon EKS. Kebijakan ini mencakup izin berikut:

Manajemen Driver CSI:

- Buat, baca, perbarui, dan hapus driver CSI, khusus untuk penyimpanan blok.

Manajemen Volume:

- Buat daftar, tonton, buat, perbarui, tambal, dan hapus volume persisten.
- Buat daftar, tonton, dan perbarui klaim volume persisten.
- Patch status klaim volume persisten.

Interaksi Node dan Pod:

- Baca informasi node dan pod.
- Kelola acara yang terkait dengan operasi penyimpanan.

Kelas dan Atribut Penyimpanan:

- Baca kelas penyimpanan dan node CSI.
- Baca kelas atribut volume.

Lampiran Volume:

- Buat daftar, tonton, dan ubah lampiran volume dan statusnya.

Operasi Snapshot:

- Kelola snapshot volume, konten snapshot, dan kelas snapshot.
- Menangani operasi untuk snapshot grup volume dan sumber daya terkait.

Kebijakan ini dirancang untuk mendukung manajemen penyimpanan blok komprehensif dalam kluster Amazon EKS yang berjalan dalam Mode Otomatis. Ini menggabungkan izin untuk berbagai operasi termasuk penyediaan, melampirkan, mengubah ukuran, dan snapshotting volume penyimpanan blok.

Amazon EKS secara otomatis membuat entri akses dengan kebijakan akses ini untuk peran IAM klaster saat Mode Otomatis diaktifkan, memastikan bahwa izin yang diperlukan tersedia agar kemampuan penyimpanan blok berfungsi dengan baik.

Amazon EKSClusterPolicy

Note

Kebijakan ini ditujukan hanya untuk peran AWS terkait layanan dan tidak dapat digunakan dengan peran yang dikelola pelanggan.

ARN– `arn:aws:eks::aws:cluster-access-policy/AmazonEKSClusterPolicy`

Kebijakan ini memberikan izin yang diperlukan untuk kemampuan manajemen komputasi Mode Otomatis Amazon EKS. Ini memungkinkan orkestrasi dan penskalaan sumber daya komputasi yang efisien dalam kluster Amazon EKS. Kebijakan ini mencakup izin berikut:

Manajemen Node:

- Membuat, membaca, memperbarui, menghapus, dan mengelola status NodePools dan NodeClaims.
- Kelola NodeClasses, termasuk pembuatan, modifikasi, dan penghapusan.

Penjadwalan dan Manajemen Sumber Daya:

- Baca akses ke pod, node, volume persisten, klaim volume persisten, pengontrol replikasi, dan ruang nama.
- Baca akses ke kelas penyimpanan, node CSI, dan lampiran volume.
- Daftar dan tonton penerapan, set daemon, set replika, dan set stateful.
- Baca anggaran gangguan pod.

Penanganan Acara:

- Buat, baca, dan kelola acara cluster.

Pencabutan Node dan Penggusuran Pod:

- Perbarui, tambal, dan hapus node.
- Buat penggusuran pod dan hapus pod bila perlu.

Manajemen Definisi Sumber Daya Kustom (CRD):

- Buat yang baru CRDs.
- Kelola spesifik CRDs yang terkait dengan manajemen node (NodeClasses NodePools, NodeClaims,, dan NodeDiagnostics).

Kebijakan ini dirancang untuk mendukung manajemen komputasi komprehensif dalam kluster Amazon EKS yang berjalan dalam Mode Otomatis. Ini menggabungkan izin untuk berbagai operasi termasuk penyediaan node, penjadwalan, penskalaan, dan optimasi sumber daya.

Amazon EKS secara otomatis membuat entri akses dengan kebijakan akses ini untuk peran IAM klaster saat Mode Otomatis diaktifkan, memastikan bahwa izin yang diperlukan tersedia agar kemampuan manajemen komputasi berfungsi dengan baik.

Amazon EKSLoad BalancingClusterPolicy

ARN– `arn:aws:eks::aws:cluster-access-policy/AmazonEKSLoadBalancingClusterPolicy`

Kebijakan ini memberikan izin yang diperlukan untuk kemampuan penyeimbangan beban Mode Otomatis Amazon EKS. Ini memungkinkan manajemen dan konfigurasi sumber daya penyeimbang beban yang efisien dalam kluster Amazon EKS. Kebijakan ini mencakup izin berikut:

Manajemen Acara dan Sumber Daya:

- Membuat dan menambal acara.
- Baca akses ke pod, node, endpoint, dan namespace.
- Perbarui status pod.

Manajemen Layanan dan Ingress:

- Manajemen penuh layanan dan statusnya.
- Kontrol komprehensif atas masuknya dan statusnya.
- Baca akses ke irisan titik akhir dan kelas ingress.

Binding Grup Sasaran:

- Membuat dan memodifikasi binding grup target dan statusnya.

- Baca akses ke parameter kelas ingress.

Manajemen Definisi Sumber Daya Kustom (CRD):

- Buat dan baca semuanya CRDs.
- Manajemen khusus `targetgroupbindings.eks.amazonaws.com` dan `ingressclassparams.eks.amazonaws.com`. CRDs

Konfigurasi Webhook:

- Membuat dan membaca mutasi dan memvalidasi konfigurasi webhook.
- Kelola `eks-load-balancing-webhook` konfigurasi.

Kebijakan ini dirancang untuk mendukung manajemen load balancing yang komprehensif dalam kluster Amazon EKS yang berjalan dalam Mode Otomatis. Ini menggabungkan izin untuk berbagai operasi termasuk eksposur layanan, perutean masuk, dan integrasi dengan layanan penyeimbangan AWS beban.

Amazon EKS secara otomatis membuat entri akses dengan kebijakan akses ini untuk peran IAM kluster saat Mode Otomatis diaktifkan, memastikan bahwa izin yang diperlukan tersedia agar kemampuan penyeimbangan beban berfungsi dengan baik.

Amazon EKSNetworking ClusterPolicy

ARN– `arn:aws:eks::aws:cluster-access-policy/AmazonEKSNetworkingClusterPolicy`

Amazon EKSNetworking ClusterPolicy

Kebijakan ini memberikan izin yang diperlukan untuk kemampuan jaringan Amazon EKS Auto Mode. Ini memungkinkan manajemen dan konfigurasi sumber daya jaringan yang efisien dalam kluster Amazon EKS. Kebijakan ini mencakup izin berikut:

Manajemen Node dan Pod:

- Baca akses ke `NodeClasses` dan status mereka.
- Baca akses ke `NodeClaims` dan status mereka.
- Baca akses ke `pod`.

Manajemen Node CNI:

- Izin untuk CNINodes dan statusnya, termasuk membuat, membaca, memperbarui, menghapus, dan menambal.

Manajemen Definisi Sumber Daya Kustom (CRD):

- Buat dan baca semuanya CRDs.
- Manajemen khusus (pembaruan, tambalan, hapus) dari CNInodes.eks.amazonaws.com CRD.

Manajemen Acara:

- Membuat dan menambal acara.

Kebijakan ini dirancang untuk mendukung manajemen jaringan yang komprehensif dalam kluster Amazon EKS yang berjalan dalam Mode Otomatis. Ini menggabungkan izin untuk berbagai operasi termasuk konfigurasi jaringan node, manajemen CNI (Container Network Interface), dan penanganan sumber daya khusus terkait.

Kebijakan ini memungkinkan komponen jaringan untuk berinteraksi dengan sumber daya terkait node, mengelola konfigurasi node khusus CNI, dan menangani sumber daya khusus yang penting untuk operasi jaringan di cluster.

Amazon EKS secara otomatis membuat entri akses dengan kebijakan akses ini untuk peran IAM klaster saat Mode Otomatis diaktifkan, memastikan bahwa izin yang diperlukan tersedia agar kemampuan jaringan berfungsi dengan baik.

EKSHybridKebijakan Amazon

Note

Kebijakan ini ditujukan hanya untuk peran AWS terkait layanan dan tidak dapat digunakan dengan peran yang dikelola pelanggan.

Kebijakan akses ini mencakup izin yang memberikan EKS akses ke node klaster. Ketika dikaitkan dengan entri akses, cakupan aksesnya biasanya adalah klaster, bukan namespace Kubernetes. Kebijakan ini digunakan oleh node hibrida Amazon EKS.

ARN– `arn:aws:eks::aws:cluster-access-policy/AmazonEKSHybridPolicy`

Grup API Kubernetes	Kubernetes NonResource URLs	Sumber daya Kubernetes	Kata kerja Kubernetes (izin)
*		nodes	list

Amazon EKSCluster InsightsPolicy

Note

Kebijakan ini ditujukan hanya untuk peran AWS terkait layanan dan tidak dapat digunakan dengan peran yang dikelola pelanggan.

ARN– `arn:aws:eks::aws:cluster-access-policy/AmazonEKSClusterInsightsPolicy`

Kebijakan ini memberikan izin hanya-baca untuk fungsionalitas Amazon EKS Cluster Insights. Kebijakan ini mencakup izin berikut:

Akses Node: - Daftar dan lihat node cluster - Baca informasi status node

DaemonSet Akses: - Baca akses ke konfigurasi kube-proxy

Kebijakan ini dikelola secara otomatis oleh layanan EKS untuk Cluster Insights. Untuk informasi selengkapnya, lihat [the section called “Wawasan cluster”](#).

Mengakses pembaruan kebijakan

Lihat detail tentang pembaruan untuk mengakses kebijakan, sejak diperkenalkan. Untuk peringatan otomatis tentang perubahan pada halaman ini, berlangganan umpan RSS di [Riwayat dokumen](#)

Perubahan	Deskripsi	Tanggal
Tambahkan kebijakan untuk EKS Cluster Insights	Publikasikan AmazonEKS ClusterInsightsPolicy	Desember 2, 2024

Perubahan	Deskripsi	Tanggal
Tambahkan kebijakan untuk Amazon EKS Hybrid	Publikasikan AmazonEKS HybridPolicy	Desember 2, 2024
Tambahkan kebijakan untuk Mode Otomatis Amazon EKS	Kebijakan akses ini memberikan izin kepada Peran IAM Cluster dan Peran IAM Node untuk memanggil Kubernetes. APIs AWS menggunakan ini untuk mengotomatiskan tugas-tugas rutin untuk penyimpanan, komputasi, dan sumber daya jaringan.	Desember 2, 2024
Menambahkan AmazonEKS AdminViewPolicy	Tambahkan kebijakan baru untuk akses tampilan yang diperluas, termasuk sumber daya seperti Rahasia.	April 23, 2024
Kebijakan akses diperkenalkan.	Amazon EKS memperkenalkan kebijakan akses.	29 Mei 2023

Ubah mode otentikasi untuk menggunakan entri akses

Untuk mulai menggunakan entri akses, Anda harus mengubah mode otentikasi cluster ke mode `API_AND_CONFIG_MAP` atau `API`. Ini menambahkan API untuk entri akses.

AWS Konsol

1. Buka [konsol Amazon EKS](#).
2. Pilih nama cluster tempat Anda ingin membuat entri akses.
3. Pilih tab Access.
4. Mode otentikasi menunjukkan mode otentikasi cluster saat ini. Jika mode mengatakan EKS API, Anda sudah dapat menambahkan entri akses dan Anda dapat melewati langkah-langkah yang tersisa.

5. Pilih Kelola akses.
6. Untuk mode otentikasi Cluster, pilih mode dengan EKS API. Perhatikan bahwa Anda tidak dapat mengubah mode autentikasi kembali ke mode yang menghapus API EKS dan entri akses.
7. Pilih Simpan perubahan. Amazon EKS mulai memperbarui cluster, status klaster berubah menjadi Memperbarui, dan perubahan dicatat di tab Riwayat Perbarui.
8. Tunggu status cluster kembali ke Aktif. Saat cluster Aktif, Anda dapat mengikuti langkah-langkah [the section called “Buat entri akses”](#) untuk menambahkan akses ke cluster untuk prinsipal IAM.

AWS CLI

1. Instal AWS CLI, seperti yang dijelaskan dalam [Menginstal](#) di Panduan Pengguna Antarmuka Baris AWS Perintah.
2. Jalankan perintah berikut. Ganti *my-cluster* dengan nama klaster Anda. Jika Anda ingin menonaktifkan ConfigMap metode secara permanen, ganti API_AND_CONFIG_MAP dengan API.

Amazon EKS mulai memperbarui cluster, status klaster berubah menjadi UPDATE, dan perubahan dicatat dalam `aws eks list-updates`.

```
aws eks update-cluster-config --name my-cluster --access-config
authenticationMode=API_AND_CONFIG_MAP
```

3. Tunggu status cluster kembali ke Aktif. Saat cluster Aktif, Anda dapat mengikuti langkah-langkah [the section called “Buat entri akses”](#) untuk menambahkan akses ke cluster untuk prinsipal IAM.

Versi platform yang diperlukan

Untuk menggunakan entri akses, klaster harus memiliki versi platform yang sama atau lebih lambat dari versi yang tercantum dalam tabel berikut, atau versi Kubernetes yang lebih lambat dari versi yang tercantum dalam tabel. Jika versi Kubernetes Anda tidak terdaftar, semua versi platform mendukung entri akses.

Versi Kubernetes	Versi platform
1.30	eks.2
1.29	eks.1

Versi Kubernetes	Versi platform
1.28	eks.6
1.27	eks.10
1.26	eks.11

Lihat informasi yang lebih lengkap di [the section called “Versi platform”](#).

Buat entri akses

Sebelum membuat entri akses, pertimbangkan hal berikut:

- Mode otentikasi yang diatur dengan benar. Lihat [the section called “Mode autentikasi”](#).
- Entri akses mencakup Nama Sumber Daya Amazon (ARN) dari satu, dan hanya satu, prinsipal IAM yang ada. Prinsipal IAM tidak dapat dimasukkan dalam lebih dari satu entri akses. Pertimbangan tambahan untuk ARN yang Anda tentukan:
 - Praktik terbaik IAM merekomendasikan untuk mengakses kluster Anda menggunakan peran IAM yang memiliki kredensi jangka pendek, daripada pengguna IAM yang memiliki kredensi jangka panjang. Untuk informasi selengkapnya, lihat [Mewajibkan pengguna manusia untuk menggunakan federasi dengan penyedia identitas untuk mengakses AWS menggunakan kredensi sementara](#) di Panduan Pengguna IAM.
 - Jika ARN adalah untuk peran IAM, itu dapat mencakup jalur. ARNs dalam `aws-auth ConfigMap` entri, tidak dapat menyertakan jalur. Misalnya, ARN Anda bisa `arn:aws:iam::<111122223333>:role/<development/apps/my-role>` atau `arn:aws:iam::<111122223333>:role/<my-role>`
 - Jika jenis entri akses adalah apa pun selain STANDARD (lihat pertimbangan selanjutnya tentang jenis), ARN harus berada di AWS akun yang sama dengan cluster Anda. Jika jenisnya STANDARD, ARN bisa berada di akun yang sama, atau berbeda, dari AWS akun tempat cluster Anda berada.
 - Anda tidak dapat mengubah prinsipal IAM setelah entri akses dibuat.
 - Jika Anda pernah menghapus prinsipal IAM dengan ARN ini, entri akses tidak dihapus secara otomatis. Kami menyarankan Anda menghapus entri akses dengan ARN untuk prinsipal IAM yang Anda hapus. Jika Anda tidak menghapus entri akses dan membuat ulang prinsipal IAM, meskipun memiliki ARN yang sama, entri akses tidak akan berfungsi. Ini karena meskipun ARN

sama untuk prinsipal IAM yang dibuat ulang, `roleID` atau `userID` (Anda dapat melihat ini dengan perintah `aws sts get-caller-identity` AWS CLI) berbeda untuk prinsipal IAM yang dibuat ulang daripada untuk prinsip IAM asli. Meskipun Anda tidak melihat kepala sekolah IAM `roleID` atau `userID` untuk entri akses, Amazon EKS menyimpannya dengan entri akses.

- Setiap entri akses memiliki tipe. Jenis entri akses tergantung pada jenis sumber daya yang terkait dengannya, dan tidak menentukan izin. Jika Anda tidak menentukan jenis, Amazon EKS secara otomatis menyetel tipe ke STANDARD
- EC2_LINUX- Untuk peran IAM yang digunakan dengan Linux atau Bottlerocket node yang dikelola sendiri
- EC2_WINDOWS- Untuk peran IAM yang digunakan dengan node Windows yang dikelola sendiri
- FARGATE_LINUX- Untuk peran IAM yang digunakan dengan AWS Fargate (Fargate)
- HYBRID_LINUX- Untuk peran IAM yang digunakan dengan node hybrid
- STANDARD- Jenis default jika tidak ada yang ditentukan
- EC2- Untuk kelas simpul khusus Mode Otomatis EKS. Untuk informasi selengkapnya, lihat [the section called “Buat entri akses kelas node”](#).
- Anda tidak dapat mengubah jenis setelah entri akses dibuat.
- Tidak perlu membuat entri akses untuk peran IAM yang digunakan untuk grup node terkelola atau profil Fargate. EKS akan membuat entri akses (jika diaktifkan), atau memperbarui peta konfigurasi autentikasi (jika entri akses tidak tersedia)
- Jika jenis entri aksesnya STANDARD, Anda dapat menentukan nama pengguna untuk entri akses. Jika Anda tidak menentukan nilai untuk nama pengguna, Amazon EKS menetapkan salah satu nilai berikut untuk Anda, tergantung pada jenis entri akses dan apakah prinsipal IAM yang Anda tentukan adalah peran IAM atau pengguna IAM. Kecuali Anda memiliki alasan khusus untuk menentukan nama pengguna Anda sendiri, sebaiknya jangan tentukan nama pengguna dan biarkan Amazon EKS membuatnya secara otomatis untuk Anda. Jika Anda menentukan nama pengguna Anda sendiri:
 - Itu tidak bisa dimulai dengan `system:`, `eks:`, `aws:`, `amazon:`, atau `iam:`.
 - Jika nama pengguna untuk peran IAM, kami sarankan Anda menambahkan `{{SessionName}}` atau `{{SessionNameRaw}}` ke akhir nama pengguna Anda. Jika Anda menambahkan salah satu `{{SessionName}}` atau `{{SessionNameRaw}}` ke nama pengguna Anda, nama pengguna harus menyertakan titik dua sebelum `{{SessionName}}`. Ketika peran ini diasumsikan, nama sesi AWS STS yang ditentukan saat mengasumsikan peran secara otomatis diteruskan ke cluster dan akan muncul di CloudTrail log. Misalnya, Anda tidak dapat memiliki nama pengguna `john{{SessionName}}`. Nama pengguna harus `:john{{SessionName}}`

atau `john{{SessionName}}`. Usus besar hanya harus sebelumnya `{{SessionName}}`. Nama pengguna yang dihasilkan oleh Amazon EKS dalam tabel berikut mencakup ARN. Karena ARN termasuk titik dua, ia memenuhi persyaratan ini. Titik dua tidak diperlukan jika Anda tidak menyertakan `{{SessionName}}` nama pengguna Anda. Perhatikan bahwa `{{SessionName}}` dalam karakter khusus “@” diganti dengan “-” di nama sesi. `{{SessionNameRaw}}` menyimpan semua karakter khusus dalam nama sesi.

Tipe utama IAM	Tipe	Nilai nama pengguna yang ditetapkan secara otomatis oleh Amazon EKS
Pengguna	STANDARD	ARN pengguna. Contoh: <code>arn:aws:iam::<111122223333>:user/<my-user></code>
Peran	STANDARD	STS ARN dari peran ketika diasumsikan. Amazon EKS <code>{{SessionName}}</code> menambahkan peran tersebut. Contoh: <code>arn:aws:sts::<111122223333>:assumed-role/<my-role>/{{SessionName}}</code> Jika ARN peran yang Anda tentukan berisi jalur, Amazon EKS menghapusnya di nama pengguna yang dihasilkan.
Peran	EC2_LINUX atau EC2_Windows	<code>system:node:{{EC2PrivateDNSName}}</code>

Tipe utama IAM	Tipe	Nilai nama pengguna yang ditetapkan secara otomatis oleh Amazon EKS
Peran	FARGATE_LINUX	system:node:{{SessionName}}
Peran	HYBRID_LINUX	system:node:{{SessionName}}

Anda dapat mengubah nama pengguna setelah entri akses dibuat.

- Jika jenis entri akses adalah `STANDARD`, dan Anda ingin menggunakan otorisasi Kubernetes RBAC, Anda dapat menambahkan satu atau beberapa nama grup ke entri akses. Setelah Anda membuat entri akses, Anda dapat menambahkan dan menghapus nama grup. Agar prinsipal IAM memiliki akses ke objek Kubernetes di kluster Anda, Anda harus membuat dan mengelola objek otorisasi berbasis peran Kubernetes (RBAC). Buat `Kubernetes RoleBinding` atau `ClusterRoleBinding` objek pada kluster Anda yang menentukan nama grup sebagai `for.subject.kind: Group`. Kubernetes mengotorisasi akses utama IAM ke objek kluster apa pun yang telah Anda tentukan dalam `Kubernetes Role` atau `ClusterRole` objek yang juga Anda tentukan dalam pengikatan Anda. `roleRef` Jika Anda menentukan nama grup, sebaiknya Anda terbiasa dengan objek Kubernetes role-based authorization (RBAC). Untuk informasi selengkapnya, lihat [Menggunakan Otorisasi RBAC](#) dalam dokumentasi Kubernetes.

Important

Amazon EKS tidak mengonfirmasi bahwa objek Kubernetes RBAC apa pun yang ada di kluster Anda menyertakan nama grup apa pun yang Anda tentukan. Misalnya, jika Anda membuat entri akses untuk grup yang saat ini tidak ada, EKS akan membuat grup alih-alih mengembalikan kesalahan.

Alih-alih, atau sebagai tambahan, Kubernetes mengotorisasi akses utama IAM ke objek Kubernetes di kluster Anda, Anda dapat mengaitkan kebijakan akses Amazon EKS ke entri akses. Amazon EKS mengotorisasi prinsipal IAM untuk mengakses objek Kubernetes di kluster Anda dengan izin dalam kebijakan akses. Anda dapat membuat cakupan izin kebijakan akses ke ruang nama Kubernetes yang Anda tentukan. Penggunaan kebijakan akses tidak mengharuskan Anda

untuk mengelola objek Kubernetes RBAC. Untuk informasi selengkapnya, lihat [the section called “Kebijakan akses asosiasi”](#).

- Jika Anda membuat entri akses dengan tipe EC2_LINUX atau EC2_Windows, prinsipal IAM yang membuat entri akses harus memiliki `iam:PassRole` izin. Untuk informasi selengkapnya, lihat [Memberikan izin pengguna untuk meneruskan peran ke AWS layanan](#) di Panduan Pengguna IAM.
- Mirip dengan [perilaku IAM](#) standar, pembuatan dan pembaruan entri akses pada akhirnya konsisten, dan mungkin memerlukan beberapa detik untuk menjadi efektif setelah panggilan API awal berhasil dikembalikan. Anda harus merancang aplikasi Anda untuk memperhitungkan potensi penundaan ini. Kami menyarankan agar Anda tidak menyertakan entri akses yang dibuat atau diperbarui di jalur kode ketersediaan tinggi yang penting dari aplikasi Anda. Sebaliknya, buat perubahan dalam inisialisasi terpisah atau rutinitas pengaturan yang lebih jarang Anda lakukan. Selain itu, pastikan untuk memverifikasi bahwa perubahan telah dibuat merata sebelum alur kerja produksi bergantung padanya.
- Entri akses tidak mendukung [peran terkait layanan](#). Anda tidak dapat membuat entri akses di mana ARN utama adalah peran terkait layanan. Anda dapat mengidentifikasi peran terkait layanan dengan ARN mereka, yang ada dalam format. `arn:aws:iam::*:role/aws-service-role/*`

Anda dapat membuat entri akses menggunakan AWS Management Console atau AWS CLI.

AWS Management Console

1. Buka [konsol Amazon EKS](#).
2. Pilih nama cluster tempat Anda ingin membuat entri akses.
3. Pilih tab Access.
4. Pilih Buat entri akses.
5. Untuk prinsipal IAM, pilih peran atau pengguna IAM yang ada. Praktik terbaik IAM merekomendasikan untuk mengakses kluster Anda menggunakan peran IAM yang memiliki kredensi jangka pendek, daripada pengguna IAM yang memiliki kredensi jangka panjang. Untuk informasi selengkapnya, lihat [Mewajibkan pengguna manusia untuk menggunakan federasi dengan penyedia identitas untuk mengakses AWS menggunakan kredensi sementara](#) di Panduan Pengguna IAM.
6. Untuk Type, jika entri akses adalah untuk peran node yang digunakan untuk EC2 node Amazon yang dikelola sendiri, pilih EC2 Linux atau EC2 Windows. Jika tidak, terima default (Standar).

7. Jika Jenis yang Anda pilih adalah Standar dan Anda ingin menentukan Nama Pengguna, masukkan nama pengguna.
8. Jika Tipe yang Anda pilih adalah Standar dan Anda ingin menggunakan otorisasi Kubernetes RBAC untuk prinsipal IAM, tentukan satu atau beberapa nama untuk Grup. Jika Anda tidak menentukan nama grup apa pun dan ingin menggunakan otorisasi Amazon EKS, Anda dapat mengaitkan kebijakan akses di langkah selanjutnya, atau setelah entri akses dibuat.
9. (Opsional) Untuk Tag, tetapkan label ke entri akses. Misalnya, untuk membuatnya lebih mudah untuk menemukan semua sumber daya dengan tag yang sama.

10. Pilih Berikutnya.

11. Pada halaman Add access policy, jika tipe yang Anda pilih adalah Standard dan Anda ingin Amazon EKS mengotorisasi prinsipal IAM untuk memiliki izin ke objek Kubernetes di kluster Anda, selesaikan langkah-langkah berikut. Jika tidak, pilih Selanjutnya.

- a. Untuk nama Kebijakan, pilih kebijakan akses. Anda tidak dapat melihat izin kebijakan akses, tetapi mereka menyertakan izin serupa dengan yang ada di objek yang dihadapi pengguna Kubernetes. `ClusterRole` Untuk informasi selengkapnya, lihat [Peran yang dihadapi pengguna dalam dokumentasi](#) Kubernetes.
- b. Pilih salah satu opsi berikut:
 - Cluster — Pilih opsi ini jika Anda ingin Amazon EKS mengotorisasi prinsipal IAM untuk memiliki izin dalam kebijakan akses untuk semua objek Kubernetes di kluster Anda.
 - Namespace Kubernetes - Pilih opsi ini jika Anda ingin Amazon EKS mengotorisasi prinsipal IAM untuk memiliki izin dalam kebijakan akses untuk semua objek Kubernetes di namespace Kubernetes tertentu di kluster Anda. Untuk Namespace, masukkan nama namespace Kubernetes di kluster Anda. Jika Anda ingin menambahkan ruang nama tambahan, pilih Tambahkan namespace baru dan masukkan nama namespace.
- c. Jika Anda ingin menambahkan kebijakan tambahan, pilih Tambah kebijakan. Anda dapat membuat cakupan setiap kebijakan secara berbeda, tetapi Anda dapat menambahkan setiap kebijakan hanya sekali.
- d. Pilih Berikutnya.

12. Tinjau konfigurasi untuk entri akses Anda. Jika ada yang terlihat salah, pilih Sebelumnya untuk kembali melalui langkah-langkah dan memperbaiki kesalahan. Jika konfigurasi sudah benar, pilih Buat.

AWS CLI

1. Instal AWS CLI, seperti yang dijelaskan dalam [Menginstal](#) di Panduan Pengguna Antarmuka Baris AWS Perintah.
2. Untuk membuat entri akses Anda dapat menggunakan salah satu contoh berikut untuk membuat entri akses:

- Buat entri akses untuk grup node Amazon EC2 Linux yang dikelola sendiri. Ganti *my-cluster* dengan nama cluster Anda, *111122223333* dengan ID AWS akun Anda, dan *EKS-my-cluster-self-managed-ng-1* dengan nama [peran IAM node](#) Anda. Jika grup node Anda adalah grup node Windows, maka ganti *EC2_LINUX* dengan *EC2_Windows*.

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn arn:aws:iam::111122223333:role/EKS-my-cluster-self-managed-ng-1 --type EC2_LINUX
```

Anda tidak dapat menggunakan `--kubernetes-groups` opsi ketika Anda menentukan jenis selain `STANDARD`. Anda tidak dapat mengaitkan kebijakan akses ke entri akses ini, karena jenisnya adalah nilai selain `STANDARD`.

- Buat entri akses yang memungkinkan peran IAM yang tidak digunakan untuk grup node yang EC2 dikelola sendiri Amazon, yang Anda inginkan Kubernetes untuk mengotorisasi akses ke klaster Anda. Ganti *my-cluster* dengan nama klaster Anda, *111122223333* dengan ID AWS akun Anda, dan *my-role* dengan nama peran IAM Anda. Ganti *Viewers* dengan nama grup yang telah Anda tentukan di Kubernetes RoleBinding atau ClusterRoleBinding objek di klaster Anda.

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn arn:aws:iam::111122223333:role/my-role --type STANDARD --user Viewers --kubernetes-groups Viewers
```

- Buat entri akses yang memungkinkan pengguna IAM untuk mengautentikasi ke cluster Anda. Contoh ini diberikan karena hal ini dimungkinkan, meskipun praktik terbaik IAM merekomendasikan untuk mengakses klaster Anda menggunakan peran IAM yang memiliki kredensi jangka pendek, daripada pengguna IAM yang memiliki kredensial jangka panjang. Untuk informasi selengkapnya, lihat [Mewajibkan pengguna manusia untuk menggunakan federasi dengan penyedia identitas untuk mengakses AWS menggunakan kredensi sementara](#) di Panduan Pengguna IAM.

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn arn:aws:iam::111122223333:user/my-user --type STANDARD --username my-user
```

Jika Anda ingin pengguna ini memiliki lebih banyak akses ke kluster Anda daripada izin di peran penemuan API Kubernetes, maka Anda perlu mengaitkan kebijakan akses ke entri akses, karena `--kubernetes-groups` opsi tersebut tidak digunakan. Untuk informasi selengkapnya, lihat [the section called “Kebijakan akses asosiasi”](#) dan [peran penemuan API](#) dalam dokumentasi Kubernetes.

Perbarui entri akses

Anda dapat memperbarui entri akses menggunakan AWS Management Console atau AWS CLI.

AWS Management Console

1. Buka [konsol Amazon EKS](#).
2. Pilih nama cluster tempat Anda ingin membuat entri akses.
3. Pilih tab Access.
4. Pilih entri akses yang ingin Anda perbarui.
5. Pilih Edit.
6. Untuk Username, Anda dapat mengubah nilai yang ada.
7. Untuk Grup, Anda dapat menghapus nama grup yang ada atau menambahkan nama grup baru. Jika nama grup berikut ada, jangan hapus mereka: `system:nodes` atau `system:bootstrappers`. Menghapus grup ini dapat menyebabkan cluster Anda berfungsi dengan tidak benar. Jika Anda tidak menentukan nama grup apa pun dan ingin menggunakan otorisasi Amazon EKS, kaitkan [kebijakan akses](#) di langkah selanjutnya.
8. Untuk Tag, Anda dapat menetapkan label ke entri akses. Misalnya, untuk membuatnya lebih mudah untuk menemukan semua sumber daya dengan tag yang sama. Anda juga dapat menghapus tag yang ada.
9. Pilih Simpan perubahan.
10. Jika Anda ingin mengaitkan kebijakan akses ke entri, lihat [the section called “Kebijakan akses asosiasi”](#).

AWS CLI

1. Instal AWS CLI, seperti yang dijelaskan dalam [Menginstal](#) di Panduan Pengguna Antarmuka Baris AWS Perintah.
2. Untuk memperbarui entri akses Ganti *my-cluster* dengan nama kluster Anda, *111122223333* dengan ID AWS akun Anda, dan *EKS-my-cluster-my-namespace-Viewers* dengan nama peran IAM.

```
aws eks update-access-entry --cluster-name my-cluster --principal-arn arn:aws:iam::111122223333:role/EKS-my-cluster-my-namespace-Viewers --kubernetes-groups Viewers
```

Anda tidak dapat menggunakan `--kubernetes-groups` opsi jika jenis entri akses adalah nilai selain `STANDARD`. Anda juga tidak dapat mengaitkan kebijakan akses ke entri akses dengan jenis selain `STANDARD`.

Hapus entri akses

Jika Anda menemukan bahwa Anda menghapus entri akses karena kesalahan, Anda selalu dapat membuatnya kembali. Jika entri akses yang Anda hapus dikaitkan dengan kebijakan akses apa pun, asosiasi akan dihapus secara otomatis. Anda tidak perlu memisahkan kebijakan akses dari entri akses sebelum menghapus entri akses.

Anda dapat menghapus entri akses menggunakan AWS Management Console atau AWS CLI.

AWS Management Console

1. Buka [konsol Amazon EKS](#).
2. Pilih nama kluster tempat Anda ingin menghapus entri akses.
3. Pilih tab Access.
4. Dalam daftar entri Access, pilih entri akses yang ingin Anda hapus.
5. Pilih Hapus.
6. Di kotak dialog konfirmasi, pilih Hapus.

AWS CLI

1. Instal AWS CLI, seperti yang dijelaskan dalam [Menginstal](#) di Panduan Pengguna Antarmuka Baris AWS Perintah.
2. Untuk menghapus entri akses Ganti *my-cluster* dengan nama kluster Anda, **111122223333** dengan ID AWS akun Anda, dan *my-role* dengan nama peran IAM yang tidak lagi ingin Anda akses ke kluster Anda.

```
aws eks delete-access-entry --cluster-name my-cluster --principal-arn arn:aws:iam::111122223333:role/my-role
```

Berikan akses kepada pengguna IAM ke Kubernetes dengan a ConfigMap

Important

`aws-auth` ConfigMap itu sudah usang. Untuk metode yang direkomendasikan untuk mengelola akses ke Kubernetes APIs, lihat. [the section called “Berikan izin”](#)

Akses ke kluster Anda menggunakan [prinsipal IAM](#) diaktifkan oleh [AWS IAM Authenticator for Kubernetes](#), yang berjalan di bidang kontrol Amazon EKS. Authenticator mendapatkan informasi konfigurasinya dari file. `aws-auth` ConfigMap Untuk semua `aws-auth` ConfigMap pengaturan, lihat [Format Konfigurasi Lengkap](#) aktif GitHub.

Tambahkan prinsipal IAM ke kluster Amazon EKS Anda

Saat Anda membuat kluster Amazon EKS, [prinsipal IAM](#) yang membuat kluster secara otomatis diberikan `system:masters` izin dalam konfigurasi kontrol akses berbasis peran (RBAC) kluster di bidang kontrol Amazon EKS. Prinsipal ini tidak muncul dalam konfigurasi yang terlihat, jadi pastikan untuk melacak prinsipal mana yang awalnya membuat cluster. Untuk memberikan prinsipal IAM tambahan kemampuan untuk berinteraksi dengan kluster Anda, edit bagian `aws-auth` ConfigMap dalam Kubernetes dan buat Kubernetes `rolebinding` atau `clusterrolebinding` dengan nama yang Anda tentukan di dalamnya. `group aws-auth` ConfigMap

Note

Untuk informasi selengkapnya tentang konfigurasi kontrol akses berbasis peran Kubernetes (RBAC), lihat [Menggunakan](#) Otorisasi RBAC dalam dokumentasi Kubernetes.

1. Tentukan kredensial mana yang `kubectl` digunakan untuk mengakses kluster Anda. Di komputer Anda, Anda dapat melihat kredensial mana yang `kubectl` digunakan dengan perintah berikut. Ganti `~/.kube/config` dengan jalur ke kubeconfig file Anda jika Anda tidak menggunakan jalur default.

```
cat ~/.kube/config
```

Contoh output adalah sebagai berikut.

```
[...]
contexts:
- context:
    cluster: my-cluster.region-code.eksctl.io
    user: admin@my-cluster.region-code.eksctl.io
    name: admin@my-cluster.region-code.eksctl.io
current-context: admin@my-cluster.region-code.eksctl.io
[...]
```

Dalam contoh keluaran sebelumnya, kredensial untuk pengguna bernama `admin` dikonfigurasi untuk kluster bernama `my-cluster`. Jika ini adalah pengguna yang membuat cluster, maka ia sudah memiliki akses ke cluster Anda. Jika bukan pengguna yang membuat cluster, maka Anda perlu menyelesaikan langkah-langkah yang tersisa untuk mengaktifkan akses cluster untuk prinsipal IAM lainnya. [Praktik terbaik IAM](#) menyarankan agar Anda memberikan izin untuk peran, bukan pengguna. Anda dapat melihat prinsipal lain mana yang saat ini memiliki akses ke cluster Anda dengan perintah berikut:

```
kubectl describe -n kube-system configmap/aws-auth
```

Contoh output adalah sebagai berikut.

```
Name:      aws-auth
Namespace: kube-system
```

```

Labels:      <none>
Annotations: <none>

Data
====
mapRoles:
----
- groups:
  - system:bootstrappers
  - system:nodes
  rolearn: arn:aws: iam::111122223333:role/my-node-role
  username: system:node:{{EC2PrivateDNSName}}

BinaryData
====

Events:  <none>

```

Contoh sebelumnya adalah default `aws-authConfigMap`. Hanya peran instance node yang memiliki akses ke cluster.

2. Pastikan Anda memiliki Kubernetes roles dan rolebindings or clusterroles dan Anda dapat memetakan clusterrolebindings prinsipal IAM. Untuk informasi selengkapnya tentang sumber daya ini, lihat [Menggunakan Otorisasi RBAC](#) di dokumentasi Kubernetes.
 - a. Lihat Kubernetes roles atau clusterroles Rolesdicakup ke anamespace, tetapi clusterroles dicakup ke cluster.

```
kubectl get roles -A
```

```
kubectl get clusterroles
```

- b. Lihat detail apa pun role atau yang clusterrole dikembalikan di keluaran sebelumnya dan konfirmasi bahwa ia memiliki izin (`rules`) yang Anda inginkan untuk dimiliki oleh prinsipal IAM Anda di cluster Anda.

Ganti *role-name* dengan role nama yang dikembalikan dalam output dari perintah sebelumnya. Ganti *kube-system* dengan namespace dari file. role

```
kubectl describe role role-name -n kube-system
```

Ganti *cluster-role-name* dengan clusterrole nama yang dikembalikan dalam output dari perintah sebelumnya.

```
kubectl describe clusterrole cluster-role-name
```

- c. Lihat Kubernetes rolebindings atau clusterrolebindings Rolebindingsdicakup ke anamespace, tetapi clusterrolebindings dicakup ke cluster.

```
kubectl get rolebindings -A
```

```
kubectl get clusterrolebindings
```

- d. Lihat detail apa pun rolebinding atau clusterrolebinding dan konfirmasi bahwa ia memiliki role atau clusterrole dari langkah sebelumnya yang terdaftar sebagai roleRef dan nama grup yang terdaftar untuksubjects.

Ganti *role-binding-name* dengan rolebinding nama yang dikembalikan dalam output dari perintah sebelumnya. Ganti *kube-system* dengan namespace darirolebinding.

```
kubectl describe rolebinding role-binding-name -n kube-system
```

Contoh output adalah sebagai berikut.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: eks-console-dashboard-restricted-access-role-binding
  namespace: default
subjects:
- kind: Group
  name: eks-console-dashboard-restricted-access-group
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: eks-console-dashboard-restricted-access-role
  apiGroup: rbac.authorization.k8s.io
```

Ganti *cluster-role-binding-name* dengan clusterrolebinding nama yang dikembalikan dalam output dari perintah sebelumnya.

```
kubectl describe clusterrolebinding cluster-role-binding-name
```

Contoh output adalah sebagai berikut.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: eks-console-dashboard-full-access-binding
subjects:
- kind: Group
  name: eks-console-dashboard-full-access-group
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: eks-console-dashboard-full-access-clusterrole
  apiGroup: rbac.authorization.k8s.io
```

3. Edit `aws-authConfigMap`. Anda dapat menggunakan alat seperti `eksctl` untuk memperbarui `ConfigMap` atau Anda dapat memperbaruinya secara manual dengan mengeditnya.

Important

Kami merekomendasikan menggunakan `eksctl`, atau alat lain, untuk mengedit `ConfigMap`. Untuk informasi tentang alat lain yang dapat Anda gunakan, lihat [Menggunakan alat untuk membuat perubahan pada `aws-authConfigMap`](#) dalam panduan praktik terbaik Amazon EKS. Format yang tidak benar `aws-auth ConfigMap` dapat menyebabkan Anda kehilangan akses ke cluster Anda.

- Lihat langkah-langkah untuk [mengedit configmap dengan](#) `eksctl`.
- Lihat langkah-langkah untuk [mengedit configmap](#) secara manual.

Edit Configmap dengan Eksctl

1. Anda memerlukan versi `0.210.0` atau yang lebih baru dari alat baris perintah yang diinstal pada perangkat Anda atau AWS CloudShell. Untuk menginstal atau memperbarui `eksctl`, lihat [Instalasi](#) dalam `eksctl` dokumentasi.

2. Lihat pemetaan saat ini di. ConfigMap Ganti *my-cluster* dengan nama kluster Anda. Ganti *region-code* dengan AWS Wilayah tempat cluster Anda berada.

```
eksctl get iamidentitymapping --cluster my-cluster --region=region-code
```

Contoh output adalah sebagai berikut.

ARN	USERNAME	GROUPS
ACCOUNT		
arn:aws: iam::111122223333:role/eksctl-my-cluster-my-nodegroup-		
NodeInstanceRole-1XLS7754U3ZPA	system:node:{{EC2PrivateDNSName}}	
	system:bootstrappers,system:nodes	

3. Tambahkan pemetaan untuk peran. Ganti *my-role* dengan nama peran Anda. Ganti *eks-console-dashboard-full-access-group* dengan nama grup yang ditentukan dalam Kubernetes RoleBinding atau objek Anda. ClusterRoleBinding Ganti *111122223333* dengan ID akun Anda. Anda dapat mengganti *admin* dengan nama apa pun yang Anda pilih.

```
eksctl create iamidentitymapping --cluster my-cluster --region=region-code \
  --arn arn:aws: iam::111122223333:role/my-role --username admin --group eks-
console-dashboard-full-access-group \
  --no-duplicate-arns
```

Important

Peran ARN tidak dapat menyertakan jalur seperti. `role/my-team/developers/my-role` Format ARN harus. `arn:aws: iam::111122223333:role/my-role` Dalam contoh ini, `my-team/developers/` perlu dihapus.

Contoh output adalah sebagai berikut.

```
[...]
2022-05-09 14:51:20 [#] adding identity "{arn-aws}iam::111122223333:role/my-role" to
auth ConfigMap
```

4. Tambahkan pemetaan untuk pengguna. [Praktik terbaik IAM](#) menyarankan agar Anda memberikan izin untuk peran, bukan pengguna. Ganti *my-user* dengan nama pengguna Anda. Ganti *eks-*

console-dashboard-restricted-access-group dengan nama grup yang ditentukan dalam Kubernetes RoleBinding atau objek Anda. ClusterRoleBinding Ganti **111122223333** dengan ID akun Anda. Anda dapat mengganti *my-user* dengan nama apa pun yang Anda pilih.

```
eksctl create iamidentitymapping --cluster my-cluster --region=region-code \
    --arn arn:aws:iam::111122223333:user/my-user --username my-user --group eks-
console-dashboard-restricted-access-group \
    --no-duplicate-arns
```

Contoh output adalah sebagai berikut.

```
[...]
2022-05-09 14:53:48 [#] adding identity "arn:aws:iam::111122223333:user/my-user" to
auth ConfigMap
```

5. Lihat pemetaan di lagi. ConfigMap

```
eksctl get iamidentitymapping --cluster my-cluster --region=region-code
```

Contoh output adalah sebagai berikut.

ARN	USERNAME	GROUPS
arn:aws:iam::111122223333:role/eksctl-my-cluster-my-nodegroup-NodeInstanceRole-1XLS7754U3ZPA	system:node:{{EC2PrivateDNSName}}	
	system:bootstrappers,system:nodes	
arn:aws:iam::111122223333:role/admin	my-role	eks-console-dashboard-full-access-group
arn:aws:iam::111122223333:user/my-user	my-user	eks-console-dashboard-restricted-access-group

Edit Configmap secara manual

1. Buka ConfigMap untuk mengedit.

```
kubectl edit -n kube-system configmap/aws-auth
```

Note

Jika Anda menerima kesalahan yang menyatakan "Error from server (NotFound): configmaps "aws-auth" not found", maka gunakan prosedur di [Terapkan aws-auth ConfigMap ke cluster Anda untuk](#) menerapkan stok. ConfigMap

2. Tambahkan prinsip IAM Anda ke. ConfigMap Grup IAM bukanlah prinsipal IAM, sehingga tidak dapat ditambahkan ke. ConfigMap
 - Untuk menambahkan peran IAM (misalnya, untuk [pengguna federasi](#)): Tambahkan detail peran ke mapRoles bagianConfigMap, di bawah. data Tambahkan bagian ini jika belum ada di dalam file. Setiap masuk mendukung parameter berikut:
 - rolearn: ARN IAM role untuk menambahkan. Nilai ini tidak dapat menyertakan jalur. Misalnya, Anda tidak dapat menentukan ARN seperti. `arn:aws: iam::111122223333:role/my-team/developers/role-name` ARN harus sebagai gantinya. `arn:aws: iam::111122223333:role/role-name`
 - nama pengguna: Nama pengguna dalam Kubernetes untuk memetakan ke dalam IAM role.
 - grup: Grup atau daftar grup Kubernetes untuk memetakan peran. Grup dapat berupa grup default, atau grup yang ditentukan dalam clusterrolebinding ataurolebinding. Untuk informasi selengkapnya, lihat [Peran default dan binding peran](#) dalam dokumentasi Kubernetes.
 - Untuk menambahkan pengguna IAM: [Praktik terbaik IAM](#) menyarankan Anda memberikan izin ke peran, bukan pengguna. Tambahkan detail pengguna ke mapUsers bagianConfigMap, di bawahdata. Tambahkan bagian ini jika belum ada di dalam file. Setiap entri mendukung parameter berikut:
 - userarn: ARN pengguna IAM untuk ditambahkan.
 - nama pengguna: nama pengguna dalam Kubernetes untuk memetakan ke pengguna IAM.
 - grup: Grup, atau daftar grup Kubernetes untuk memetakan pengguna. Grup dapat berupa grup default, atau grup yang ditentukan dalam clusterrolebinding ataurolebinding. Untuk informasi selengkapnya, lihat [Peran default dan binding peran](#) dalam dokumentasi Kubernetes.
3. Misalnya, blok YAMAL berikut berisi:
 - mapRolesBagian yang memetakan instance node IAM ke grup Kubernetes sehingga node dapat mendaftarkan diri mereka sendiri dengan klaster dan peran my-console-viewer-role IAM yang dipetakan ke grup Kubernetes yang dapat melihat semua sumber daya Kubernetes

untuk semua cluster. Untuk daftar izin grup IAM dan Kubernetes yang diperlukan untuk peran IAM, lihat. `my-console-viewer-role` [the section called “Izin yang diperlukan”](#)

- `mapUsers` Bagian yang memetakan pengguna admin IAM dari AWS akun default ke grup `system:masters` Kubernetes dan `my-user` pengguna dari AWS akun lain yang dipetakan ke grup Kubernetes yang dapat melihat sumber daya Kubernetes untuk namespace tertentu. Untuk daftar izin grup IAM dan Kubernetes yang diperlukan untuk pengguna IAM, lihat. `my-user` [the section called “Izin yang diperlukan”](#)

Tambahkan atau hapus baris seperlunya dan ganti semua *example values* dengan nilai Anda sendiri.

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file
# will be
# reopened with the relevant failures.
#
apiVersion: v1
data:
  mapRoles: |
    - groups:
      - system:bootstrappers
      - system:nodes
      rolearn: arn:aws:iam::111122223333:role/my-role
      username: system:node:{{EC2PrivateDNSName}}
    - groups:
      - eks-console-dashboard-full-access-group
      rolearn: arn:aws:iam::111122223333:role/my-console-viewer-role
      username: my-console-viewer-role
  mapUsers: |
    - groups:
      - system:masters
      userarn: arn:aws:iam::111122223333:user/admin
      username: admin
    - groups:
      - eks-console-dashboard-restricted-access-group
      userarn: arn:aws:iam::444455556666:user/my-user
      username: my-user
```

4. Simpan file dan keluar dari editor teks Anda.

Terapkan **aws-authConfigMap** ke cluster Anda

Secara otomatis **aws-auth ConfigMap** dibuat dan diterapkan ke cluster Anda ketika Anda membuat grup node terkelola atau ketika Anda membuat grup node menggunakan `eksctl`. Ini awalnya dibuat untuk memungkinkan node bergabung dengan cluster Anda, tetapi Anda juga menggunakan ini **ConfigMap** untuk menambahkan akses kontrol akses berbasis peran (RBAC) ke prinsip-prinsip IAM. Jika Anda telah meluncurkan node yang dikelola sendiri dan belum menerapkannya **aws-auth ConfigMap** ke cluster Anda, Anda dapat melakukannya dengan prosedur berikut.

1. Periksa untuk melihat apakah Anda sudah menerapkan **aws-authConfigMap**.

```
kubectl describe configmap -n kube-system aws-auth
```

Jika Anda menerima kesalahan yang menyatakan "Error from server (NotFound): configmaps "aws-auth" not found", maka lanjutkan dengan langkah-langkah berikut untuk menerapkan **aws-authConfigMap**.

2. Unduh, edit, dan terapkan peta konfigurasi AWS autentikator.

- a. Unduh peta konfigurasi.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/
aws-auth-cm.yaml
```

- b. Dalam **aws-auth-cm.yaml** file, atur `roleARN` ke Amazon Resource Name (ARN) dari peran IAM yang terkait dengan node Anda. Anda dapat melakukan ini dengan editor teks, atau dengan mengganti **my-node-instance-role** dan menjalankan perintah berikut:

```
sed -i.bak -e 's|<ARN of instance role (not instance profile)>|my-node-instance-
role|' aws-auth-cm.yaml
```

Jangan mengubah baris lain dalam file ini.

Important

Peran ARN tidak dapat menyertakan jalur seperti. `role/my-team/developers/my-role` Format ARN harus. `arn:aws:iam::111122223333:role/my-role` Dalam contoh ini, `my-team/developers/` perlu dihapus.

Anda dapat memeriksa output AWS CloudFormation tumpukan untuk grup node Anda dan mencari nilai berikut:

- InstanceRoleARN — Untuk grup node yang dibuat dengan eksctl
 - NodeInstanceRole— Untuk grup simpul yang dibuat dengan AWS CloudFormation templat penjual Amazon EKS di AWS Management Console
- c. Terapkan konfigurasi. Perintah ini mungkin memerlukan waktu beberapa menit untuk diselesaikan.

```
kubectl apply -f aws-auth-cm.yaml
```

 Note

Jika Anda menerima kesalahan otorisasi atau jenis sumber daya, lihat [the section called “Tidak sah atau akses ditolak \(kubectl\)”](#) di topik pemecahan masalah.

3. Perhatikan status simpul Anda dan tunggu sampai simpul mencapai Status Ready.

```
kubectl get nodes --watch
```

Masukkan `Ctrl+C` untuk kembali ke prompt shell.

Beri pengguna akses ke Kubernetes dengan penyedia OIDC eksternal

Amazon EKS mendukung penggunaan penyedia identitas OpenID Connect (OIDC) sebagai metode untuk mengautentikasi pengguna ke klaster Anda. Penyedia identitas OIDC dapat digunakan dengan, atau sebagai alternatif untuk AWS Identity and Access Management (IAM). Untuk informasi selengkapnya tentang penggunaan IAM, lihat [the section called “Akses API Kubernetes”](#). Setelah mengonfigurasi autentikasi ke klaster Anda, Anda dapat membuat roles dan clusterroles Kubernetes untuk memberikan izin ke dalam peran, dan kemudian mengikat peran ke dalam identitas menggunakan rolebindings dan clusterrolebindings Kubernetes. Untuk informasi selengkapnya, lihat [Menggunakan Otorisasi RBAC](#) dalam dokumentasi Kubernetes.

- Anda dapat mengaitkan satu penyedia identitas OIDC ke klaster Anda.

- Kubernetes tidak menyediakan penyedia identitas OIDC. Anda dapat menggunakan penyedia identitas OIDC publik yang sudah ada, atau Anda dapat menjalankan penyedia identitas milik Anda sendiri. Untuk daftar penyedia tersertifikasi, lihat [OpenID Certification](#) di situs OpenID.
- URL penerbit penyedia identitas OIDC harus dapat diakses publik, sehingga Amazon EKS dapat menemukan kunci penandatanganan. Amazon EKS tidak mendukung penyedia identitas OIDC dengan sertifikat yang ditandatangani sendiri.
- Anda tidak dapat menonaktifkan autentikasi IAM ke kluster Anda, karena masih diperlukan untuk menggabungkan node ke cluster.
- Cluster Amazon EKS masih harus dibuat oleh [prinsipal AWS IAM](#), bukan pengguna penyedia identitas OIDC. Ini karena pembuat cluster berinteraksi dengan Amazon EKS APIs, bukan APIs Kubernetes.
- Pengguna yang diautentikasi penyedia identitas OIDC tercantum dalam log audit kluster jika CloudWatch log dihidupkan untuk bidang kontrol. Untuk informasi selengkapnya, lihat [the section called “Mengaktifkan atau menonaktifkan log bidang kontrol”](#).
- Anda tidak dapat masuk ke akun AWS Management Console dengan akun dari penyedia OIDC. Anda hanya [the section called “Akses sumber daya kluster”](#) dapat masuk ke akun Identity and Access Management AWS Management Console dengan akun AWS Identity and Access Management.

Kaitkan penyedia identitas OIDC

Sebelum Anda dapat mengaitkan penyedia identitas OIDC dengan kluster Anda, Anda memerlukan informasi berikut dari penyedia Anda:

URL Penerbit

URL penyedia identitas OIDC yang memungkinkan server API menemukan kunci penandatanganan publik untuk memverifikasi token. URL harus dimulai dengan `https://` dan harus sesuai dengan `iss` klaim dalam token ID OIDC penyedia. Sesuai dengan standar OIDC, komponen jalur diperbolehkan tetapi parameter kueri tidak. Biasanya URL hanya terdiri dari nama host, seperti <https://server.example.org> atau <https://example.com>. URL ini harus mengarah ke level di bawah `.well-known/openid-configuration` dan harus dapat diakses oleh publik melalui internet.

ID Klien (juga dikenal sebagai audiens)

ID untuk aplikasi klien yang membuat permintaan otentikasi ke penyedia identitas OIDC.

Anda dapat mengaitkan penyedia identitas menggunakan `eksctl` atau AWS Management Console.

Kaitkan penyedia identitas menggunakan `eksctl`

1. Buat file bernama *associate-identity-provider.yaml* dengan isi berikut ini. Ganti *example values* dengan milik Anda sendiri. Nilai-nilai dalam bagian `identityProviders` diperoleh dari penyedia identitas OIDC Anda. Nilai hanya diperlukan untuk `name`, `type`, `issuerUrl`, dan pengaturan `clientId` di bawah `identityProviders`.

```
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: your-region-code

identityProviders:
  - name: my-provider
    type: oidc
    issuerUrl: https://example.com
    clientId: kubernetes
    usernameClaim: email
    usernamePrefix: my-username-prefix
    groupsClaim: my-claim
    groupsPrefix: my-groups-prefix
    requiredClaims:
      string: string
    tags:
      env: dev
```

Important

Jangan tentukan `system:`, atau bagian mana pun dari string itu, untuk `groupsPrefix` atau `usernamePrefix`.

2. Buat penyedia.

```
eksctl associate identityprovider -f associate-identity-provider.yaml
```

3. Untuk digunakan `kubectl` untuk bekerja dengan cluster dan penyedia identitas OIDC Anda, lihat [Menggunakan kubectl di dokumentasi Kubernetes](#).

Kaitkan penyedia identitas menggunakan AWS Konsol

1. Buka [konsol Amazon EKS](#).
2. Pilih klaster Anda, lalu pilih tab Access.
3. Di bagian Penyedia Identitas OIDC, pilih* Penyedia Identitas Rekan*.
4. Pada halaman Penyedia Identitas Associate OIDC, masukkan atau pilih opsi berikut, dan kemudian pilih Associate.
 - Untuk Nama, masukkan nama unik untuk penyedia.
 - Untuk Penerbit URL, masukkan URL untuk penyedia Anda. URL ini harus dapat diakses melalui internet.
 - Untuk ID Klien, masukkan ID klien penyedia identitas OIDC (juga dikenal sebagai audiens).
 - Untuk Klaim nama pengguna, masukkan klaim untuk digunakan sebagai nama pengguna.
 - Untuk klaim Grup, masukkan klaim yang akan digunakan sebagai grup pengguna.
 - (Opsional) Pilih Opsi lanjutan, masukkan atau pilih informasi berikut.
 - Prefiks nama pengguna – Masukkan prefiks untuk ditambahkan ke klaim nama pengguna. Prefiks diawali dengan nama pengguna klaim untuk mencegah bentrokan dengan nama yang sudah ada. Jika Anda tidak memberikan nilai, dan nama pengguna adalah nilai selain email, prefiks default ke dalam nilai untuk Penerbit URL. Anda dapat menggunakan -nilai untuk menonaktifkan semua prefiks. Jangan tentukan `system:` atau bagian mana pun dari string itu.
 - Prefiks grup – Masukkan prefiks untuk ditambahkan ke klaim grup. Prefiks ditambahkan ke klaim grup untuk mencegah bentrokan dengan nama yang sudah ada (seperti `system: groups`). Misalnya, `oidc:` nilai membuat nama grup seperti `oidc:engineering` dan `oidc:infra`. Jangan tentukan `system:` atau bagian mana pun dari string itu..
 - Klaim yang diperlukan – Pilih Tambahkan klaim dan masukkan satu atau beberapa pasangan nilai kunci yang menjelaskan klaim yang diperlukan dalam token ID klien. Pasangan tersebut menjelaskan klaim yang diperlukan dalam Token ID. Jika diatur, setiap klaim diverifikasi untuk ada dalam token ID dengan nilai yang cocok.
 - a. Untuk digunakan `kubectl` untuk bekerja dengan cluster dan penyedia identitas OIDC Anda, lihat [Menggunakan kubectl di dokumentasi Kubernetes](#).

Contoh kebijakan IAM

Jika Anda ingin mencegah penyedia identitas OIDC dikaitkan dengan kluster, buat dan kaitkan kebijakan IAM berikut ke dalam akun IAM Amazon EKS administrator Anda. Untuk informasi selengkapnya, lihat [Membuat kebijakan IAM](#) dan [Menambahkan izin identitas IAM](#) di Panduan Pengguna dan [Tindakan](#) IAM di Referensi Otorisasi Layanan.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "denyOIDC",
      "Effect": "Deny",
      "Action": [
        "eks:AssociateIdentityProviderConfig"
      ],
      "Resource": "arn:aws:eks:us-west-2.amazonaws.com:111122223333:cluster/*"
    },
    {
      "Sid": "eksAdmin",
      "Effect": "Allow",
      "Action": [
        "eks:*"
      ],
      "Resource": "*"
    }
  ]
}
```

Contoh kebijakan berikut memungkinkan asosiasi penyedia identitas OIDC jika `clientId` adalah `kubernetes` dan `issuerUrl` adalah [https://cognito-idp.us-west-2amazonaws.com/](https://cognito-idp.us-west-2.amazonaws.com/)
* .

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCognitoOnly",
      "Effect": "Deny",
      "Action": "eks:AssociateIdentityProviderConfig",
      "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/my-instance",
    }
  ]
}
```

```

        "Condition": {
            "StringNotLikeIfExists": {
                "eks:issuerUrl": "https://cognito-idp.us-west-2.amazonaws.com/*"
            }
        },
        {
            "Sid": "DenyOtherClients",
            "Effect": "Deny",
            "Action": "eks:AssociateIdentityProviderConfig",
            "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/my-instance",
            "Condition": {
                "StringNotEquals": {
                    "eks:clientId": "kubernetes"
                }
            }
        },
        {
            "Sid": "AllowOthers",
            "Effect": "Allow",
            "Action": "eks:*",
            "Resource": "*"
        }
    ]
}

```

Putuskan penyedia identitas OIDC dari klaster Anda

Jika Anda memisahkan penyedia identitas OIDC dari klaster Anda, pengguna yang disertakan dalam penyedia tidak dapat lagi mengakses klaster. Namun, Anda masih dapat mengakses cluster dengan [prinsipal IAM](#).

1. Buka [konsol Amazon EKS](#).
2. Pada bagian Penyedia Identitas OIDC, pilih Pisahkan, masukkan nama penyedia identitas, dan kemudian pilih Disassociate.

Lihat sumber daya Kubernetes di AWS Management Console

Anda dapat melihat sumber daya Kubernetes yang diterapkan ke klaster Anda dengan file. AWS Management Console [Anda tidak dapat melihat sumber daya Kubernetes dengan CLI AWS atau eksctl. Untuk melihat sumber daya Kubernetes menggunakan alat baris perintah, gunakan kubectl.](#)

Note

Untuk melihat tab Resources dan bagian Nodes pada tab Compute di AWS Management Console, [prinsipal IAM](#) yang Anda gunakan harus memiliki izin IAM dan Kubernetes tertentu. Untuk informasi selengkapnya, lihat [the section called “Izin yang diperlukan”](#).

1. Buka [konsol Amazon EKS](#).
2. Dalam daftar Clusters, pilih cluster yang berisi resource Kubernetes yang ingin Anda lihat.
3. Pilih tab Sumber Daya.
4. Pilih grup tipe sumber daya yang ingin Anda lihat sumber daya, seperti Beban kerja. Anda melihat daftar jenis sumber daya di grup itu.
5. Pilih jenis sumber daya, seperti Deployment, di grup Beban kerja. Anda melihat deskripsi tipe sumber daya, tautan ke dokumentasi Kubernetes untuk informasi selengkapnya tentang tipe sumber daya, dan daftar sumber daya dari jenis tersebut yang digunakan di klaster Anda. Jika daftar kosong, maka tidak ada sumber daya dari jenis itu yang diterapkan ke cluster Anda.
6. Pilih sumber daya untuk melihat informasi lebih lanjut tentangnya. Coba contoh berikut:
 - Pilih grup Beban kerja, pilih jenis sumber daya Deployment, lalu pilih sumber daya coredns. Ketika Anda memilih sumber daya, Anda berada dalam tampilan Terstruktur, secara default. Untuk beberapa jenis sumber daya, Anda melihat bagian Pod dalam tampilan Terstruktur. Bagian ini mencantumkan Pod yang dikelola oleh beban kerja. Anda dapat memilih Pod apa pun yang terdaftar untuk melihat informasi tentang Pod. Tidak semua jenis sumber daya menampilkan informasi dalam Tampilan Terstruktur. Jika Anda memilih tampilan Raw di sudut kanan atas halaman untuk sumber daya, Anda akan melihat respons JSON lengkap dari API Kubernetes untuk sumber daya.
 - Pilih grup Cluster dan kemudian pilih jenis sumber daya Node. Anda melihat daftar semua node di cluster Anda. Node dapat berupa [jenis simpul Amazon EKS](#). Ini adalah daftar yang sama yang Anda lihat di bagian Nodes ketika Anda memilih tab Compute untuk cluster Anda. Pilih sumber daya node dari daftar. Dalam tampilan Terstruktur, Anda juga melihat bagian Pod. Bagian ini menunjukkan kepada Anda semua Pod yang berjalan pada node.

Izin yang diperlukan

Untuk melihat tab Resources dan bagian Nodes pada tab Compute di AWS Management Console, [prinsipal IAM](#) yang Anda gunakan harus memiliki izin IAM dan Kubernetes minimum tertentu.

Selesaikan langkah-langkah berikut untuk menetapkan izin yang diperlukan untuk kepala sekolah IAM Anda.

1. Pastikan bahwa `eks:AccessKubernetesApi`, dan izin IAM lain yang diperlukan untuk melihat sumber daya Kubernetes, ditetapkan ke prinsipal IAM yang Anda gunakan. Untuk informasi selengkapnya tentang cara mengedit izin untuk prinsipal IAM, lihat [Mengontrol akses untuk prinsipal di Panduan Pengguna IAM](#). Untuk informasi selengkapnya tentang cara mengedit izin untuk peran, lihat [Memodifikasi kebijakan izin peran \(konsol\)](#) di Panduan Pengguna IAM.

Contoh kebijakan berikut mencakup izin yang diperlukan bagi prinsipal untuk melihat sumber daya Kubernetes untuk semua kluster di akun Anda. Ganti **111122223333** dengan ID AWS akun Anda.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "eks:ListFargateProfiles",
        "eks:DescribeNodegroup",
        "eks:ListNodegroups",
        "eks:ListUpdates",
        "eks:AccessKubernetesApi",
        "eks:ListAddons",
        "eks:DescribeCluster",
        "eks:DescribeAddonVersions",
        "eks:ListClusters",
        "eks:ListIdentityProviderConfigs",
        "iam:ListRoles"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "ssm:GetParameter",
      "Resource": "arn:aws:ssm:*:111122223333:parameter/*"
    }
  ]
}
```

Untuk melihat node dalam [cluster yang terhubung](#), [peran IAM konektor Amazon EKS](#) harus dapat meniru prinsipal di cluster. Hal ini memungkinkan [Amazon EKS Connector](#) untuk memetakan prinsipal ke pengguna Kubernetes.

2. Buat Kubernetes `rolebinding` atau `clusterrolebinding` yang terikat pada Kubernetes atau yang memiliki izin yang diperlukan untuk melihat `role` sumber `clusterrole` daya Kubernetes. Untuk mempelajari lebih lanjut tentang peran Kubernetes dan binding peran, lihat [Menggunakan Otorisasi RBAC](#) dalam dokumentasi Kubernetes. Anda dapat menerapkan salah satu manifes berikut ke kluster Anda yang membuat `role` dan `rolebinding` atau a `clusterrole` dan `clusterrolebinding` dengan izin Kubernetes yang diperlukan:

Lihat sumber daya Kubernetes di semua ruang nama

- Nama grup dalam file tersebut adalah `eks-console-dashboard-full-access-group`. Terapkan manifes ke cluster Anda dengan perintah berikut:

```
kubectl apply -f https://s3.us-west-2.amazonaws.com/amazon-eks/docs/eks-console-full-access.yaml
```

Melihat sumber daya Kubernetes di namespace tertentu

- Namespace dalam file ini adalah `default`. Nama grup dalam file tersebut adalah `eks-console-dashboard-restricted-access-group`. Terapkan manifes ke cluster Anda dengan perintah berikut:

```
kubectl apply -f https://s3.us-west-2.amazonaws.com/amazon-eks/docs/eks-console-restricted-access.yaml
```

Jika Anda perlu mengubah nama grup Kubernetes, namespace, izin, atau konfigurasi lainnya dalam file, maka unduh file tersebut dan edit sebelum menerapkannya ke cluster Anda:

- a. Unduh file dengan salah satu perintah berikut:

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/docs/eks-console-full-access.yaml
```

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/docs/eks-console-restricted-access.yaml
```

- b. Edit file seperlunya.
- c. Terapkan manifes ke cluster Anda dengan salah satu perintah berikut:

```
kubectl apply -f eks-console-full-access.yaml
```

```
kubectl apply -f eks-console-restricted-access.yaml
```

3. Petakan [prinsipal IAM](#) ke pengguna atau grup Kubernetes di `aws-auth ConfigMap` Anda dapat menggunakan alat seperti `eksctl` untuk memperbarui `ConfigMap` atau Anda dapat memperbaruinya secara manual dengan mengeditnya.

Important

Kami merekomendasikan menggunakan `eksctl`, atau alat lain, untuk mengedit `ConfigMap`. Untuk informasi tentang alat lain yang dapat Anda gunakan, lihat [Menggunakan alat untuk membuat perubahan pada `aws-auth ConfigMap`](#) dalam panduan praktik terbaik Amazon EKS. Format yang tidak benar `aws-auth ConfigMap` dapat menyebabkan Anda kehilangan akses ke cluster Anda.

Edit dengan `eksctl`

1. Anda memerlukan versi `0.210.0` atau yang lebih baru dari alat baris perintah yang diinstal pada perangkat Anda atau AWS CloudShell. Untuk menginstal atau memperbarui `eksctl`, lihat [Instalasi](#) dalam `eksctl` dokumentasi.
2. Lihat pemetaan saat ini di `ConfigMap` Ganti *my-cluster* dengan nama kluster Anda. Ganti *region-code* dengan AWS Wilayah tempat kluster Anda berada.

```
eksctl get iamidentitymapping --cluster my-cluster --region=region-code
```

Contoh output adalah sebagai berikut.

ARN	USERNAME	GROUPS
ACCOUNT		
arn:aws:iam::111122223333:role/eksctl-my-cluster-my-nodegroup-		
NodeInstanceRole-1XLS7754U3ZPA	system:node:{{EC2PrivateDNSName}}	
system:bootstrappers,system:nodes		

3. Tambahkan pemetaan untuk peran. Contoh ini mengasumsikan bahwa Anda melampirkan izin IAM pada langkah pertama ke peran bernama *my-console-viewer-role* Ganti *111122223333* dengan ID akun Anda.

```
eksctl create iamidentitymapping \  
  --cluster my-cluster \  
  --region=region-code \  
  --arn arn:aws: iam::111122223333:role/my-console-viewer-role \  
  --group eks-console-dashboard-full-access-group \  
  --no-duplicate-arns
```

 Important

Peran ARN tidak dapat menyertakan jalur seperti. *role/my-team/developers/my-role* Format ARN harus. *arn:aws: iam::111122223333:role/my-role* Dalam contoh ini, *my-team/developers/* perlu dihapus.

Contoh output adalah sebagai berikut.

```
[...]
2022-05-09 14:51:20 [#] adding identity "arn:aws: iam::111122223333:role/my-console-viewer-role" to auth ConfigMap
```

4. Tambahkan pemetaan untuk pengguna. [Praktik terbaik IAM](#) menyarankan agar Anda memberikan izin untuk peran, bukan pengguna. Contoh ini mengasumsikan bahwa Anda melampirkan izin IAM pada langkah pertama ke pengguna bernama *my-user* Ganti *111122223333* dengan ID akun Anda.

```
eksctl create iamidentitymapping \  
  --cluster my-cluster \  
  --region=region-code \  
  --arn arn:aws: iam::111122223333:user/my-user \  
  --group eks-console-dashboard-restricted-access-group \  
  --no-duplicate-arns
```

Contoh output adalah sebagai berikut.

```
[...]
```

```
2022-05-09 14:53:48 [#] adding identity "arn:aws: iam::111122223333:user/my-user" to
auth ConfigMap
```

5. Lihat pemetaan di lagi. ConfigMap

```
eksctl get iamidentitymapping --cluster my-cluster --region=region-code
```

Contoh output adalah sebagai berikut.

ARN	USERNAME	GROUPS
arn:aws: iam::111122223333:role/eksctl-my-cluster-my-nodegroup-NodeInstanceRole-1XLS7754U3ZPA	system:node:{{EC2PrivateDNSName}}	
arn:aws: iam::111122223333:role/my-console-viewer-role		eks-console-dashboard-full-access-group
arn:aws: iam::111122223333:user/my-user		eks-console-dashboard-restricted-access-group

Edit ConfigMap secara manual

Untuk informasi selengkapnya tentang menambahkan pengguna atau peran ke dalam aws-authConfigMap, lihat [the section called “Tambahkan prinsipal IAM ke kluster Amazon EKS Anda”](#).

1. Buka aws-auth ConfigMap untuk mengedit.

```
kubectl edit -n kube-system configmap/aws-auth
```

2. Tambahkan pemetaan ke aws-authConfigMap, tetapi jangan mengganti pemetaan yang ada. Contoh berikut menambahkan pemetaan antara [prinsipal IAM](#) dengan izin yang ditambahkan pada langkah pertama dan grup Kubernetes yang dibuat pada langkah sebelumnya:

- **my-console-viewer-role** Peran dan eks-console-dashboard-full-access-group.
- **my-user** Pengguna dan eks-console-dashboard-restricted-access-group.

Contoh ini mengasumsikan bahwa Anda melampirkan izin IAM pada langkah pertama ke peran bernama *my-console-viewer-role* dan nama pengguna. *my-user* Ganti *111122223333* dengan ID AWS akun Anda.

```
apiVersion: v1
data:
  mapRoles: |
    - groups:
      - eks-console-dashboard-full-access-group
      rolearn: arn:aws: iam::111122223333:role/my-console-viewer-role
      username: my-console-viewer-role
  mapUsers: |
    - groups:
      - eks-console-dashboard-restricted-access-group
      userarn: arn:aws: iam::111122223333:user/my-user
      username: my-user
```

Important

Peran ARN tidak dapat menyertakan jalur seperti. *role/my-team/developers/my-console-viewer-role* Format ARN harus. *arn:aws:iam::111122223333:role/my-console-viewer-role* Dalam contoh ini, *my-team/developers/* perlu dihapus.

3. Simpan file, dan tutup editor teks Anda.

Connect kubectl ke kluster EKS dengan membuat file kubeconfig

Dalam topik ini, Anda membuat kubeconfig file untuk kluster Anda (atau memperbarui yang sudah ada).

Alat `kubectl` baris perintah menggunakan informasi konfigurasi dalam kubeconfig file untuk berkomunikasi dengan server API cluster. Untuk informasi selengkapnya, lihat [Mengatur Akses Cluster Menggunakan File kubeconfig](#) dalam dokumentasi Kubernetes.

Amazon EKS menggunakan `aws eks get-token` perintah `kubectl` untuk otentikasi cluster. Secara default, AWS CLI menggunakan kredensial yang sama yang dikembalikan dengan perintah berikut:

```
aws sts get-caller-identity
```

- Sebuah klaster Amazon EKS yang sudah ada. Untuk menyebarkan satu, lihat [Memulai](#).
- Alat baris kubect1 perintah diinstal pada perangkat Anda atau AWS CloudShell. Versinya bisa sama dengan atau hingga satu versi minor lebih awal atau lebih lambat dari versi Kubernetes dari klaster Anda. Misalnya, jika versi cluster Anda 1.29, Anda dapat menggunakan kubect1 versi 1.28, 1.29, atau 1.30 dengan itu. Untuk menginstal atau memutakhirkan kubect1, lihat [the section called “Siapkan kubect1 dan eksctl”](#).
- Versi 2.12.3 atau yang lebih baru atau versi 1.27.160 atau yang lebih baru dari AWS Command Line Interface (AWS CLI) diinstal dan dikonfigurasi pada perangkat Anda atau. AWS CloudShell Untuk memeriksa versi Anda saat ini, gunakan `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Package manager seperti yumapt-get,, atau Homebrew untuk macOS seringkali merupakan beberapa versi di belakang versi terbaru CLI. AWS Untuk menginstal versi terbaru, lihat [Menginstal](#) dan [Konfigurasi cepat dengan aws configure](#) di Panduan Pengguna Antarmuka Baris AWS Perintah. Versi AWS CLI yang diinstal AWS CloudShell mungkin juga beberapa versi di belakang versi terbaru. Untuk memperbaruinya, lihat [Menginstal AWS CLI ke direktori home Anda](#) di AWS CloudShell Panduan Pengguna.
- Pengguna IAM atau peran dengan izin untuk menggunakan tindakan eks:DescribeCluster API untuk klaster yang Anda tentukan. Untuk informasi selengkapnya, lihat [the section called “Kebijakan berbasis identitas”](#). Jika Anda menggunakan identitas dari penyedia OpenID Connect Anda sendiri untuk mengakses klaster Anda, lihat [Menggunakan kubect1 di dokumentasi Kubernetes](#) untuk membuat atau memperbarui file Anda. kube config

Buat kubeconfig file secara otomatis

- Versi 2.12.3 atau yang lebih baru atau versi 1.27.160 atau yang lebih baru dari AWS Command Line Interface (AWS CLI) diinstal dan dikonfigurasi pada perangkat Anda atau. AWS CloudShell Untuk memeriksa versi Anda saat ini, gunakan `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Package manager seperti yumapt-get,, atau Homebrew untuk macOS seringkali merupakan beberapa versi di belakang versi terbaru CLI. AWS Untuk menginstal versi terbaru, lihat [Menginstal](#) dan [Konfigurasi cepat dengan aws configure](#) di Panduan Pengguna Antarmuka Baris AWS Perintah. Versi AWS CLI yang diinstal AWS CloudShell mungkin juga beberapa versi di belakang versi terbaru. Untuk memperbaruinya, lihat [Menginstal AWS CLI ke direktori home Anda](#) di AWS CloudShell Panduan Pengguna.

- Izin untuk menggunakan tindakan `eks:DescribeCluster` API untuk kluster yang Anda tentukan. Untuk informasi selengkapnya, lihat [the section called “Kebijakan berbasis identitas”](#).
1. Buat atau perbarui file `kubeconfig` untuk kluster Anda. Ganti *region-code* dengan AWS Region tempat cluster Anda berada dan ganti *my-cluster* dengan nama cluster Anda.

```
aws eks update-kubeconfig --region region-code --name my-cluster
```

Secara default, file konfigurasi yang dihasilkan dibuat di `kubeconfig` jalur default (`.kube`) di direktori home Anda atau digabungkan dengan `config` file yang ada di lokasi tersebut. Anda dapat menentukan jalur lain dengan opsi `--kubeconfig`.

Anda dapat menentukan ARN IAM role dengan pilihan `--role-arn` untuk digunakan dalam autentikasi ketika Anda mengeluarkan perintah `kubectl`. Jika tidak, [prinsipal IAM](#) dalam rantai kredensi AWS CLI atau SDK default Anda akan digunakan. Anda dapat melihat identitas AWS CLI atau SDK default Anda dengan menjalankan perintah `aws sts get-caller-identity`

Untuk semua opsi yang tersedia, jalankan `aws eks update-kubeconfig help` perintah atau lihat [update-kubeconfig](#) di CLI Command Reference. AWS

2. Uji konfigurasi Anda.

```
kubectl get svc
```

Contoh output adalah sebagai berikut.

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
svc/kubernetes	ClusterIP	10.100.0.1	<none>	443/TCP	1m

Jika Anda menerima kesalahan otorisasi atau jenis sumber daya, lihat [the section called “Tidak sah atau akses ditolak \(kubectl\)”](#) di topik pemecahan masalah.

Berikan akses beban kerja Kubernetes untuk AWS menggunakan Akun Layanan Kubernetes

A Kubernetes service account provides an identity for processes that run in a Pod. For more information see [Mengelola Akun Layanan](#) in the Kubernetes documentation. If your Pod needs access to AWS services, you can map the service account to an AWS Identity and Access

Management identity to grant that access. For more information, see [the section called “Kredensial dengan IRSA”](#) or [the section called “Identitas Pod”](#).

Token akun layanan

[BoundServiceAccountTokenVolume](#) Fitur ini diaktifkan secara default dalam versi Kubernetes. Fitur ini meningkatkan keamanan token akun layanan dengan memungkinkan beban kerja yang berjalan di Kubernetes untuk meminta token web JSON yang bersifat audience, time, dan key bound. Token akun layanan memiliki masa kedaluwarsa satu jam. Di versi Kubernetes sebelumnya, token tidak memiliki kedaluwarsa. Ini berarti bahwa klien yang mengandalkan token ini harus menyegarkan token dalam waktu satu jam. [Klien Kubernetes](#) berikut SDKs menyegarkan token secara otomatis dalam jangka waktu yang diperlukan:

- Versi Go 0.15.7 dan yang lebih baru
- Versi 12.0.0 Python dan yang lebih baru
- Versi Java 9.0.0 dan yang lebih baru
- JavaScript versi 0.10.3 dan yang lebih baru
- Cabang Ruby master
- Versi Haskell 0.3.0.0
- Versi C# 7.0.5 dan yang lebih baru

Jika beban kerja Anda menggunakan versi klien sebelumnya, maka Anda harus memperbaruinya. Untuk mengaktifkan kelancaran migrasi klien ke token akun layanan terikat waktu yang lebih baru, Kubernetes menambahkan periode kedaluwarsa yang diperpanjang ke token akun layanan selama satu jam default. Untuk cluster Amazon EKS, periode kedaluwarsa yang diperpanjang adalah 90 hari. Server API Kubernetes klaster Amazon EKS Anda menolak permintaan dengan token yang berusia lebih dari 90 hari. Kami menyarankan Anda memeriksa aplikasi Anda dan dependensinya untuk memastikan bahwa klien Kubernetes sama atau lebih lambat dari versi yang SDKs tercantum sebelumnya.

Saat server API menerima permintaan dengan token yang berumur lebih dari satu jam, server API akan menganotasi peristiwa log audit API. `annotations.authentication.k8s.io/stale-token` Nilai anotasi terlihat seperti contoh berikut:

```
subject: system:serviceaccount:common:fluent-bit, seconds after warning threshold: 4185802.
```

Jika klaster Anda mengaktifkan [pencatatan bidang kontrol](#), maka anotasi ada di log audit. Anda dapat menggunakan kueri [Wawasan CloudWatch Log](#) berikut untuk mengidentifikasi semua Pod di klaster Amazon EKS Anda yang menggunakan token basi:

```
fields @timestamp
|filter @logStream like /kube-apiserver-audit/
|filter @message like /seconds after warning threshold/
|parse @message "subject: *, seconds after warning threshold:*\" as subject,
elapsedtime
```

`subject` ini mengacu pada akun layanan yang digunakan Pod. `elapsedtime` ini menunjukkan waktu yang telah berlalu (dalam detik) setelah membaca token terbaru. Permintaan ke server API ditolak ketika `elapsedtime` melebihi 90 hari (7.776.000 detik). Anda harus secara proaktif memperbarui SDK klien Kubernetes aplikasi Anda untuk menggunakan salah satu versi yang terdaftar sebelumnya yang secara otomatis menyegarkan token. Jika token akun layanan yang digunakan mendekati 90 hari dan Anda tidak memiliki cukup waktu untuk memperbarui versi SDK klien Anda sebelum token kedaluwarsa, maka Anda dapat menghentikan Pod yang ada dan membuat yang baru. Ini menghasilkan refetching token akun layanan, memberi Anda tambahan 90 hari untuk memperbarui versi klien Anda. SDKs

Jika Pod adalah bagian dari penerapan, cara yang disarankan untuk menghentikan Pod sambil menjaga ketersediaan tinggi adalah dengan melakukan peluncuran dengan perintah berikut. Ganti *my-deployment* dengan nama penerapan Anda.

```
kubectl rollout restart deployment/my-deployment
```

Pengaya klaster

Add-on klaster berikut telah diperbarui untuk menggunakan klien Kubernetes SDKs yang secara otomatis mengisi ulang token akun layanan. Sebaiknya pastikan bahwa versi yang terdaftar, atau versi yang lebih baru, diinstal pada cluster Anda.

- Plugin Amazon VPC CNI untuk Kubernetes dan plugin pembantu metrik versi dan yang lebih baru. 1.8.0 Untuk memeriksa versi Anda saat ini atau memperbaruinya, lihat [the section called “Amazon VPC CNI”](#) dan [cni-metrics-helper](#).
- Versi CoreDNS dan 1.8.4 yang lebih baru. Untuk memeriksa versi Anda saat ini atau memperbaruinya, lihat [the section called “CoreDNS”](#).

- AWS Load Balancer Controller versi 2.0.0 dan yang lebih baru. Untuk memeriksa versi Anda saat ini atau memperbaruinya, lihat [the section called “AWS Pengontrol Load Balancer”](#).
- kube-proxy Versi saat ini. Untuk memeriksa versi Anda saat ini atau memperbaruinya, lihat [the section called “kube-proxy”](#).
- AWS untuk versi Fluent Bit 2.25.0 atau yang lebih baru. Untuk memperbarui versi Anda saat ini, lihat [Rilis](#) di GitHub.
- [Fluentd image versi 1.14.6-1.2 atau yang lebih baru dan plugin filter Fluentd untuk metadata Kubernetes versi 2.11.1 atau yang lebih baru.](#)

Memberikan izin AWS Identity and Access Management untuk beban kerja di kluster Amazon Elastic Kubernetes Service

Amazon EKS menyediakan dua cara untuk memberikan izin AWS Identity and Access Management ke beban kerja yang berjalan di kluster Amazon EKS: peran IAM untuk akun layanan, dan EKS Pod Identities.

Peran IAM untuk akun layanan

Peran IAM untuk akun layanan (IRSA) mengonfigurasi aplikasi Kubernetes yang berjalan AWS dengan izin IAM berbutir halus untuk mengakses berbagai sumber daya lain seperti bucket AWS Amazon S3, tabel Amazon DynamoDB, dan banyak lagi. Anda dapat menjalankan beberapa aplikasi bersama-sama dalam kluster Amazon EKS yang sama, dan memastikan setiap aplikasi hanya memiliki set izin minimum yang diperlukan. IRSA dibangun untuk mendukung berbagai opsi penerapan Kubernetes yang didukung oleh AWS seperti Amazon EKS, Amazon EKS Anywhere, Red Hat OpenShift Service on AWS, dan cluster Kubernetes yang dikelola sendiri di instans Amazon. EC2 Dengan demikian, IRSA dibangun menggunakan AWS layanan dasar seperti IAM, dan tidak mengambil ketergantungan langsung pada layanan Amazon EKS dan EKS API. Untuk informasi selengkapnya, lihat [the section called “Kredensial dengan IRSA”](#).

Identitas EKS Pod

EKS Pod Identity menawarkan kepada administrator kluster alur kerja yang disederhanakan untuk mengautentikasi aplikasi untuk mengakses berbagai AWS sumber daya lain seperti bucket Amazon S3, tabel Amazon DynamoDB, dan banyak lagi. EKS Pod Identity hanya untuk EKS, dan sebagai hasilnya, ini menyederhanakan bagaimana administrator kluster dapat mengkonfigurasi aplikasi Kubernetes untuk mendapatkan izin IAM. Izin ini sekarang dapat dengan mudah dikonfigurasi dengan lebih sedikit langkah langsung melalui AWS Management

Console, EKS API, dan AWS CLI, dan tidak ada tindakan apa pun yang harus dilakukan di dalam cluster di objek Kubernetes mana pun. Administrator kluster tidak perlu beralih antara layanan EKS dan IAM, atau menggunakan operasi IAM istimewa untuk mengonfigurasi izin yang diperlukan oleh aplikasi Anda. Peran IAM sekarang dapat digunakan di beberapa cluster tanpa perlu memperbarui kebijakan kepercayaan peran saat membuat cluster baru. Kredensi IAM yang disediakan oleh EKS Pod Identity mencakup tag sesi peran, dengan atribut seperti nama cluster, namespace, nama akun layanan. Tag sesi peran memungkinkan administrator untuk membuat peran tunggal yang dapat bekerja di seluruh akun layanan dengan mengizinkan akses ke AWS sumber daya berdasarkan tag yang cocok. Untuk informasi selengkapnya, lihat [the section called “Identitas Pod”](#).

Membandingkan Identitas Pod EKS dan IRSA

Pada tingkat tinggi, baik EKS Pod Identity dan IRSA memungkinkan Anda untuk memberikan izin IAM untuk aplikasi yang berjalan pada kluster Kubernetes. Tetapi mereka pada dasarnya berbeda dalam cara Anda mengonfigurasinya, batas yang didukung, dan fitur yang diaktifkan. Di bawah ini, kami membandingkan beberapa aspek kunci dari kedua solusi.

Atribut	Identitas Pod EKS	IRSA
Ekstensibilitas peran	Anda harus mengatur setiap peran sekali untuk membangun kepercayaan dengan prinsipal layanan Amazon EKS yang baru diperkenalkan. <code>pods.eks.amazonaws.com</code> Setelah langkah satu kali ini, Anda tidak perlu memperbarui kebijakan kepercayaan peran setiap kali digunakan di kluster baru.	Anda harus memperbarui kebijakan kepercayaan peran IAM dengan titik akhir penyedia OIDC cluster EKS baru setiap kali Anda ingin menggunakan peran tersebut di cluster baru.
Skalabilitas cluster	EKS Pod Identity tidak mengharuskan pengguna untuk menyiapkan penyedia	Setiap kluster EKS memiliki URL penerbit OpenID Connect (OIDC) yang terkait dengannya. Untuk

Atribut	Identitas Pod EKS	IRSA
	IAM OIDC, jadi batasan ini tidak berlaku.	menggunakan IRSA, penyedia OpenID Connect yang unik perlu dibuat untuk setiap cluster EKS di IAM. IAM memiliki batas global default 100 penyedia OIDC untuk setiap akun. AWS Jika Anda berencana untuk memiliki lebih dari 100 kluster EKS untuk setiap AWS akun dengan IRSA, maka Anda akan mencapai batas penyedia IAM OIDC.
Skalabilitas peran	EKS Pod Identity tidak mengharuskan pengguna untuk menentukan hubungan kepercayaan antara peran IAM dan akun layanan dalam kebijakan kepercayaan, sehingga batasan ini tidak berlaku.	Dalam IRSA, Anda mendefinisikan hubungan kepercayaan antara peran IAM dan akun layanan dalam kebijakan kepercayaan peran. Secara default, panjang ukuran kebijakan kepercayaan adalah 2048. Ini berarti bahwa Anda biasanya dapat mendefinisikan 4 hubungan kepercayaan dalam satu kebijakan kepercayaan. Meskipun Anda bisa mendapatkan batas panjang kebijakan kepercayaan yang meningkat, Anda biasanya terbatas pada maksimal 8 hubungan kepercayaan dalam satu kebijakan kepercayaan.

Atribut	Identitas Pod EKS	IRSA
Peran dapat digunakan kembali	AWS Kredensi sementara STS yang disediakan oleh EKS Pod Identity mencakup tag sesi peran, seperti nama cluster, namespace, nama akun layanan. Tag sesi peran memungkinkan administrator untuk membuat peran IAM tunggal yang dapat digunakan dengan beberapa akun layanan, dengan izin efektif yang berbeda, dengan mengizinkan akses ke AWS sumber daya berdasarkan tag yang dilampirkan padanya. Ini juga disebut Attribute-based Access Control (ABAC). Untuk informasi selengkapnya, lihat the section called “Akses Grant Pod” .	AWS Tag sesi STS tidak didukung. Anda dapat menggunakan kembali peran antar klaster, tetapi setiap pod menerima semua izin peran tersebut.
Lingkungan yang didukung	EKS Pod Identity hanya tersedia di Amazon EKS.	IRSA dapat digunakan seperti Amazon EKS, Amazon EKS Anywhere, Red Hat OpenShift Service on AWS, dan cluster Kubernetes yang dikelola sendiri di instans Amazon. EC2
Versi EKS didukung	Semua versi cluster EKS yang didukung. Untuk versi platform tertentu, lihat the section called “EKS Pod Identity versi cluster” .	Semua versi cluster EKS yang didukung.

IAM role untuk akun layanan

Aplikasi dalam kontainer Pod dapat menggunakan AWS SDK atau AWS CLI untuk membuat permintaan API AWS ke layanan AWS menggunakan izin Identity and Access Management (IAM). Aplikasi harus menandatangani permintaan AWS API mereka dengan AWS kredensialnya. Peran IAM untuk akun layanan (IRSA) menyediakan kemampuan untuk mengelola kredensial untuk aplikasi Anda, mirip dengan cara profil EC2 instans Amazon memberikan kredensial ke instans Amazon. EC2 Alih-alih membuat dan mendistribusikan AWS kredensialmu ke container atau menggunakan peran EC2 instans Amazon, kamu mengaitkan peran IAM dengan akun layanan Kubernetes dan mengonfigurasi Pod kamu untuk menggunakan akun layanan. Anda tidak dapat menggunakan peran IAM untuk akun layanan dengan [kluster lokal untuk Amazon EKS di AWS Outposts](#).

Peran IAM untuk akun layanan memberikan manfaat berikut:

- Keistimewaan paling kecil — Anda dapat memberikan cakupan izin IAM ke akun layanan, dan hanya Pod yang menggunakan akun layanan tersebut yang memiliki akses ke izin tersebut. Fitur ini juga mengurangi kebutuhan akan solusi dari pihak ketiga seperti kiam atau kube2iam.
- Isolasi kredensial — Ketika akses ke [Amazon EC2 Instance Metadata Service \(IMDS\)](#) dibatasi, kontainer Pod hanya dapat mengambil kredensial untuk peran IAM yang terkait dengan akun layanan yang digunakan container. Container tidak pernah memiliki akses ke kredensial yang digunakan oleh kontainer lain di Pod lain. Jika IMDS tidak dibatasi, kontainer Pod juga memiliki akses ke [peran IAM node Amazon EKS](#) dan kontainer mungkin dapat memperoleh akses ke kredensial-kredensial peran IAM dari Pod lain pada node yang sama. Untuk informasi selengkapnya, lihat [Membatasi akses ke profil instance yang ditetapkan ke node pekerja](#).

Note

Pod yang dikonfigurasi `hostNetwork: true` akan selalu memiliki akses IMDS, tetapi CLI AWS SDKs dan akan menggunakan kredensial IRSA saat diaktifkan.

- Auditabilitas — Akses dan pencatatan peristiwa tersedia AWS CloudTrail untuk membantu memastikan audit retrospektif.

⚠ Important

Kontainer bukan batas keamanan, dan penggunaan peran IAM untuk akun layanan tidak mengubah ini. Pod yang ditetapkan ke node yang sama akan berbagi kernel dan kemungkinan sumber daya lainnya tergantung pada konfigurasi Pod Anda. Sementara Pod yang berjalan pada node terpisah akan diisolasi pada lapisan komputasi, ada aplikasi node yang memiliki izin tambahan di API Kubernetes di luar cakupan instance individual. Beberapa contohnya adalah kubelet, kube-proxy, driver penyimpanan CSI, atau aplikasi Kubernetes Anda sendiri.

Aktifkan peran IAM untuk akun layanan dengan menyelesaikan prosedur berikut:

1. [Buat penyedia IAM OIDC untuk kluster Anda — Anda hanya menyelesaikan prosedur ini satu kali untuk setiap cluster.](#)

📘 Note

Jika Anda mengaktifkan titik akhir EKS VPC, titik akhir layanan EKS OIDC tidak dapat diakses dari dalam VPC itu. Akibatnya, operasi Anda seperti membuat penyedia OIDC dengan eksctl VPC tidak akan berfungsi dan akan menghasilkan batas waktu saat mencoba meminta. <https://oidc.eks.region.amazonaws.com> Contoh pesan kesalahan berikut:

```
server cant find oidc.eks.region.amazonaws.com: NXDOMAIN
```

Untuk menyelesaikan langkah ini, Anda dapat menjalankan perintah di luar VPC, misalnya di dalam AWS CloudShell atau di komputer yang terhubung ke internet. Atau, Anda dapat membuat resolver bersyarat split-horizon di VPC, seperti Route 53 Resolver untuk menggunakan resolver yang berbeda untuk URL Penerbit OIDC dan tidak menggunakan DNS VPC untuk itu. [Untuk contoh penerusan bersyarat di CoreDNS, lihat permintaan fitur Amazon EKS di GitHub](#)

2. [Tetapkan peran IAM ke akun layanan Kubernetes](#) — Selesaikan prosedur ini untuk setiap set izin unik yang Anda inginkan untuk dimiliki aplikasi.
3. [Konfigurasi Pod untuk menggunakan akun layanan Kubernetes](#) — Selesaikan prosedur ini untuk setiap Pod yang membutuhkan akses ke layanan. AWS

4. [Gunakan IRSA dengan AWS SDK](#) — Konfirmasikan bahwa beban kerja menggunakan AWS SDK versi yang didukung dan beban kerja menggunakan rantai kredensial default.

Informasi latar belakang IAM, Kubernetes, dan OpenID Connect (OIDC)

Pada tahun 2014, AWS Identity and Access Management menambahkan dukungan untuk identitas federasi menggunakan OpenID Connect (OIDC). Fitur ini memungkinkan Anda untuk mengautentikasi panggilan AWS API dengan penyedia identitas yang didukung dan menerima token web OIDC JSON (JWT) yang valid. Anda dapat meneruskan token ini ke operasi AWS STS `AssumeRoleWithWebIdentity` API dan menerima kredensial peran sementara IAM. Anda dapat menggunakan kredensial ini untuk berinteraksi dengan AWS layanan apa pun, termasuk Amazon S3 dan DynamoDB.

Setiap token JWT ditandatangani oleh tanda tangan key pair. Kunci disajikan pada penyedia OIDC yang dikelola oleh Amazon EKS dan kunci pribadi berputar setiap 7 hari. Amazon EKS menyimpan kunci publik sampai kedaluwarsa. Jika Anda menghubungkan klien OIDC eksternal, ketahuilah bahwa Anda perlu menyegarkan kunci penandatanganan sebelum kunci publik kedaluwarsa. Pelajari cara [Mengambil kunci penandatanganan untuk memvalidasi token OIDC](#).

Kubernetes telah lama menggunakan akun layanan sebagai sistem identitas internalnya sendiri. Pod-pod dapat mengautentikasi dengan server API Kubernetes menggunakan token yang dipasang secara otomatis (yang merupakan non-OIDC JWT) yang hanya dapat divalidasi oleh server API Kubernetes. Token akun layanan lama ini tidak kedaluwarsa, dan memutar kunci penandatanganan adalah proses yang sulit. Dalam versi Kubernetes 1.12, dukungan ditambahkan untuk fitur baru. `ProjectedServiceAccountToken` Fitur ini adalah token web OIDC JSON yang juga berisi identitas akun layanan dan mendukung audiens yang dapat dikonfigurasi.

Amazon EKS menghosting titik akhir penemuan OIDC publik untuk setiap cluster yang berisi kunci penandatanganan untuk token web `ProjectedServiceAccountToken` JSON sehingga sistem eksternal, seperti IAM, dapat memvalidasi dan menerima token OIDC yang dikeluarkan oleh Kubernetes.

Buat penyedia IAM OIDC untuk kluster Anda

Cluster Anda memiliki URL penerbit [OpenID Connect](#) (OIDC) yang terkait dengannya. Untuk menggunakan peran AWS Identity and Access Management (IAM) and Access Management (IAM) untuk akun layanan, penyedia IAM OIDC harus ada untuk URL penerbit OIDC kluster Anda.

- Sebuah kluster Amazon EKS yang sudah ada. Untuk menyebarkan satu, lihat [Memulai](#).

- Versi 2.12.3 atau yang lebih baru atau versi 1.27.160 atau yang lebih baru dari AWS Command Line Interface (AWS CLI) diinstal dan dikonfigurasi pada perangkat Anda atau AWS CloudShell. Untuk memeriksa versi Anda saat ini, gunakan `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Package manager seperti `yum` atau `apt-get`, atau Homebrew untuk macOS seringkali merupakan beberapa versi di belakang versi terbaru CLI. AWS. Untuk menginstal versi terbaru, lihat [Menginstal](#) dan [Konfigurasi cepat dengan aws configure](#) di Panduan Pengguna Antarmuka Baris Perintah AWS. Versi AWS CLI yang diinstal di AWS CloudShell mungkin juga beberapa versi di belakang versi terbaru. Untuk memperbaruinya, lihat [Menginstal AWS CLI ke direktori home Anda](#) di AWS CloudShell Panduan Pengguna.
- Alat baris perintah `kubectl` diinstal pada perangkat Anda atau AWS CloudShell. Versinya bisa sama dengan atau hingga satu versi minor lebih awal atau lebih lambat dari versi Kubernetes dari kluster Anda. Misalnya, jika versi cluster Anda 1.29, Anda dapat menggunakan `kubectl` versi 1.28, 1.29, atau 1.30 dengan itu. Untuk menginstal atau memutakhirkan `kubectl`, lihat [the section called “Siapkan kubectl dan eksctl”](#).
- `kubectl` config file yang sudah ada yang berisi konfigurasi cluster Anda. Untuk membuat `kubectl` config file, lihat [the section called “Akses kluster dengan kubectl”](#).

Anda dapat membuat penyedia IAM OIDC untuk cluster Anda menggunakan `eksctl` atau AWS Management Console.

Buat penyedia OIDC (`eksctl`)

1. Versi 0.210.0 atau yang lebih baru dari alat baris perintah `eksctl` yang diinstal pada perangkat Anda atau AWS CloudShell. Untuk menginstal atau memperbarui `eksctl`, lihat [Instalasi](#) dalam `eksctl` dokumentasi.
2. Tentukan ID penerbit OIDC untuk kluster Anda.

Ambil ID penerbit OIDC cluster Anda dan simpan dalam variabel. Ganti `<my-cluster>` dengan nilai milik Anda sendiri.

```
cluster_name=<my-cluster>
oidc_id=$(aws eks describe-cluster --name $cluster_name --query
  "cluster.identity.oidc.issuer" --output text | cut -d '/' -f 5)
echo $oidc_id
```

3. Tentukan apakah penyedia IAM OIDC dengan ID penerbit kluster Anda sudah ada di akun Anda.

```
aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
```

Jika output dikembalikan, maka Anda sudah memiliki penyedia IAM OIDC untuk cluster Anda dan Anda dapat melewati langkah berikutnya. Jika tidak ada output yang dikembalikan, maka Anda harus membuat penyedia IAM OIDC untuk cluster Anda.

4. Buat penyedia identitas IAM OIDC untuk klaster Anda dengan perintah berikut.

```
eksctl utils associate-iam-oidc-provider --cluster $cluster_name --approve
```

Note

Jika Anda mengaktifkan titik akhir EKS VPC, titik akhir layanan EKS OIDC tidak dapat diakses dari dalam VPC itu. Akibatnya, operasi Anda seperti membuat penyedia OIDC dengan eksctl VPC tidak akan berfungsi dan akan menghasilkan batas waktu. Contoh pesan kesalahan berikut:

```
** server cant find oidc.eks.<region-code>.amazonaws.com: NXDOMAIN
```

Untuk menyelesaikan langkah ini, Anda dapat menjalankan perintah di luar VPC, misalnya di dalam AWS CloudShell atau di komputer yang terhubung ke internet. Atau, Anda dapat membuat resolver bersyarat split-horizon di VPC, seperti Route 53 Resolver untuk menggunakan resolver yang berbeda untuk URL Penerbit OIDC dan tidak menggunakan DNS VPC untuk itu. [Untuk contoh penerusan bersyarat di CoreDNS, lihat permintaan fitur Amazon EKS di GitHub](#)

Buat penyedia OIDC (Konsol)AWS

1. Buka [konsol Amazon EKS](#).
2. Di panel kiri, pilih Cluster, lalu pilih nama cluster Anda di halaman Clusters.
3. Di bagian Detail pada tab Ikhtisar, perhatikan nilai URL penyedia OpenID Connect.
4. Buka konsol IAM di <https://console.aws.amazon.com/iam/>.
5. Di panel navigasi kiri, pilih Penyedia Identitas di bawah Manajemen akses. Jika Penyedia tercantum dalam daftar yang cocok dengan URL untuk klaster Anda, maka Anda sudah memiliki

penyedia untuk klaster Anda. Jika penyedia tidak terdaftar yang cocok dengan URL untuk klaster Anda, maka Anda harus membuatnya.

6. Untuk membuat penyedia, pilih Tambah penyedia.
7. Untuk jenis Provider, pilih OpenID Connect.
8. Untuk URL Penyedia, masukkan URL penyedia OIDC untuk klaster Anda.
9. Untuk Audiens, masukkan `sts.amazonaws.com`.
10. (Opsional) Tambahkan tag apa pun, misalnya tag untuk mengidentifikasi cluster mana untuk penyedia ini.
11. Pilih Tambah penyedia.

Langkah selanjutnya: [the section called “Tetapkan peran IAM”](#)

Tetapkan peran IAM ke akun layanan Kubernetes

Topik ini membahas cara mengonfigurasi akun layanan Kubernetes untuk mengambil peran AWS Identity and Access Management (IAM). Pod apa pun yang dikonfigurasi untuk menggunakan akun layanan kemudian dapat mengakses AWS layanan apa pun yang perannya memiliki izin untuk diakses.

Prasyarat

- Sebuah klaster yang sudah ada. Jika Anda tidak memilikinya, Anda dapat membuatnya dengan mengikuti salah satu panduan di [Memulai](#).
- Penyedia IAM OpenID Connect (OIDC) yang sudah ada untuk klaster Anda. Untuk mengetahui apakah Anda sudah memilikinya atau cara membuatnya, lihat [the section called “Penyedia IAM OIDC”](#).
- Versi 2.12.3 atau yang lebih baru atau versi 1.27.160 atau yang lebih baru dari AWS Command Line Interface (AWS CLI) diinstal dan dikonfigurasi pada perangkat Anda atau AWS CloudShell. Untuk memeriksa versi Anda saat ini, gunakan `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Package manager seperti `yum` atau `apt-get`, atau Homebrew untuk macOS seringkali merupakan beberapa versi di belakang versi terbaru CLI AWS. Untuk menginstal versi terbaru, lihat [Menginstal](#) dan [Konfigurasi cepat dengan aws configure](#) di Panduan Pengguna Antarmuka Baris AWS Perintah. Versi AWS CLI yang diinstal AWS CloudShell mungkin juga beberapa versi di belakang versi terbaru. Untuk memperbaruinya, lihat [Menginstal AWS CLI ke direktori home Anda](#) di AWS CloudShell Panduan Pengguna.

- Alat baris `kubectl` perintah diinstal pada perangkat Anda atau AWS CloudShell. Versinya bisa sama dengan atau hingga satu versi minor lebih awal atau lebih lambat dari versi Kubernetes dari kluster Anda. Misalnya, jika versi cluster Anda `1.29`, Anda dapat menggunakan `kubectl` versi `1.28`, `1.29`, atau `1.30` dengan itu. Untuk menginstal atau memutakhirkan `kubectl`, lihat [the section called “Siapkan kubectl dan eksctl”](#).
- `kubectl` config file yang sudah ada yang berisi konfigurasi cluster Anda. Untuk membuat `kubectl` config file, lihat [the section called “Akses kluster dengan kubectl”](#).

Langkah 1: Buat Kebijakan IAM

Jika Anda ingin mengaitkan kebijakan IAM yang ada dengan peran IAM Anda, lewati ke langkah berikutnya.

1. Buat kebijakan IAM. Anda dapat membuat kebijakan sendiri, atau menyalin kebijakan AWS terkelola yang telah memberikan beberapa izin yang Anda perlukan dan menyesuaikannya dengan persyaratan spesifik Anda. Untuk informasi selengkapnya, lihat [Membuat kebijakan IAM](#) dalam Panduan Pengguna IAM.
2. Buat file yang menyertakan izin untuk AWS layanan yang ingin diakses oleh Pod. Untuk daftar semua tindakan untuk semua AWS layanan, lihat [Referensi Otorisasi Layanan](#).

Anda dapat menjalankan perintah berikut untuk membuat contoh file kebijakan yang memungkinkan akses hanya-baca ke bucket Amazon S3. Anda dapat secara opsional menyimpan informasi konfigurasi atau skrip bootstrap di bucket ini, dan container di Pod Anda dapat membaca file dari bucket dan memuatnya ke dalam aplikasi Anda. Jika Anda ingin membuat kebijakan contoh ini, salin konten berikut ke perangkat Anda. Ganti `my-pod-secrets-bucket` dengan nama bucket Anda dan jalankan perintah.

```
cat >my-policy.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::my-pod-secrets-bucket"
    }
  ]
}
```

EOF

3. Buat kebijakan IAM.

```
aws iam create-policy --policy-name my-policy --policy-document file://my-policy.json
```

Langkah 2: Buat dan kaitkan Peran IAM

Buat peran IAM dan kaitkan dengan akun layanan Kubernetes. Anda dapat menggunakan salah satu `eksctl` atau AWS CLI.

Buat dan asosiasikan peran (`eksctl`)

`eksctl` Perintah ini membuat akun layanan Kubernetes di namespace yang ditentukan, membuat peran IAM (jika tidak ada) dengan nama yang ditentukan, melampirkan ARN kebijakan IAM yang ada ke peran, dan membuat anotasi akun layanan dengan peran IAM ARN. Pastikan untuk mengganti nilai placeholder sampel dalam perintah ini dengan nilai spesifik Anda. Untuk menginstal atau memperbarui `eksctl`, lihat [Instalasi](#) dalam `eksctl` dokumentasi.

```
eksctl create iamserviceaccount --name my-service-account --namespace default --cluster  
my-cluster --role-name my-role \  
--attach-policy-arn arn:aws:iam::111122223333:policy/my-policy --approve
```

Important

Jika akun peran atau layanan sudah ada, perintah sebelumnya mungkin gagal. `eksctl` memiliki opsi berbeda yang dapat Anda berikan dalam situasi tersebut. Untuk informasi lebih lanjut jalankan `eksctl create iamserviceaccount --help`.

Buat dan asosiasikan peran (AWS CLI)

Jika Anda memiliki akun layanan Kubernetes yang ingin Anda ambil peran IAM, Anda dapat melewati langkah ini.

1. Buat akun layanan Kubernetes. Salin konten berikut ke perangkat Anda. Ganti *my-service-account* dengan nama yang Anda inginkan dan *default* dengan namespace yang berbeda, jika perlu. Jika Anda berubah *default*, namespace harus sudah ada.

```
cat >my-service-account.yaml <<EOF
apiVersion: v1
kind: ServiceAccount
metadata:
  name: my-service-account
  namespace: default
EOF
kubectl apply -f my-service-account.yaml
```

2. Setel ID AWS akun Anda ke variabel lingkungan dengan perintah berikut.

```
account_id=$(aws sts get-caller-identity --query "Account" --output text)
```

3. Tetapkan penyedia identitas OIDC cluster Anda ke variabel lingkungan dengan perintah berikut. Ganti *my-cluster* dengan nama klaster Anda.

```
oidc_provider=$(aws eks describe-cluster --name my-cluster --region $AWS_REGION --
query "cluster.identity.oidc.issuer" --output text | sed -e "s/^https:\\/\\/\\/" )
```

4. Tetapkan variabel untuk namespace dan nama akun layanan. Ganti *my-service-account* dengan akun layanan Kubernetes yang ingin Anda ambil peran. Ganti *default* dengan namespace akun layanan.

```
export namespace=default
export service_account=my-service-account
```

5. Jalankan perintah berikut untuk membuat file kebijakan kepercayaan untuk peran IAM. Jika Anda ingin mengizinkan semua akun layanan dalam namespace untuk menggunakan peran, salin konten berikut ke perangkat Anda. Ganti *StringEquals* dengan *StringLike* dan ganti *\$service_account* dengan*. Anda dapat menambahkan beberapa entri dalam *StringLike* kondisi *StringEquals* atau untuk memungkinkan beberapa akun layanan atau ruang nama mengambil peran. Untuk mengizinkan peran dari akun yang berbeda dari AWS akun yang digunakan klaster Anda untuk mengambil peran, lihat [the section called “IAM lintas akun”](#) untuk informasi selengkapnya.

```
cat >trust-relationship.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```

    "Effect": "Allow",
    "Principal": {
      "Federated": "arn:aws: iam::$account_id:oidc-provider/$oidc_provider"
    },
    "Action": "sts:AssumeRoleWithWebIdentity",
    "Condition": {
      "StringEquals": {
        "$oidc_provider:aud": "sts.amazonaws.com",
        "$oidc_provider:sub": "system:serviceaccount:$namespace:$service_account"
      }
    }
  }
]
}
EOF

```

6. Buat peran. Ganti *my-role* dengan nama untuk peran IAM Anda, dan *my-role-description* dengan deskripsi untuk peran Anda.

```
aws iam create-role --role-name my-role --assume-role-policy-document file://trust-relationship.json --description "my-role-description"
```

7. Lampirkan kebijakan IAM ke peran Anda. Ganti *my-role* dengan nama peran IAM Anda dan *my-policy* dengan nama kebijakan yang sudah ada yang Anda buat.

```
aws iam attach-role-policy --role-name my-role --policy-arn=arn:aws: iam::
$account_id:policy/my-policy
```

8. Beri anotasi akun layanan Anda dengan Nama Sumber Daya Amazon (ARN) dari peran IAM yang Anda inginkan untuk diasumsikan oleh akun layanan. Ganti *my-role* dengan nama peran IAM Anda yang ada. Misalkan Anda mengizinkan peran dari AWS akun yang berbeda dari akun tempat kluster Anda berada untuk mengambil peran pada langkah sebelumnya. Kemudian, pastikan untuk menentukan AWS akun dan peran dari akun lain. Untuk informasi selengkapnya, lihat [the section called "IAM lintas akun"](#).

```
kubectl annotate serviceaccount -n $namespace $service_account eks.amazonaws.com/
role-arn=arn:aws: iam::$account_id:role/my-role
```

9. (Opsional) [Konfigurasi titik akhir Layanan Token AWS Keamanan untuk akun layanan](#). AWS merekomendasikan menggunakan titik akhir AWS STS regional alih-alih titik akhir global. Ini mengurangi latensi, menyediakan redundansi bawaan, dan meningkatkan validitas token sesi.

Langkah 3: Konfirmasikan konfigurasi

1. Konfirmasikan bahwa kebijakan kepercayaan peran IAM telah dikonfigurasi dengan benar.

```
aws iam get-role --role-name my-role --query Role.AssumeRolePolicyDocument
```

Contoh output adalah sebagai berikut.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:oidc-provider/
oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "oidc.eks.region-code.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:default:my-service-
account",
          "oidc.eks.region-code.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com"
        }
      }
    }
  ]
}
```

2. Konfirmasikan bahwa kebijakan yang Anda lampirkan pada peran Anda di langkah sebelumnya dilampirkan pada peran tersebut.

```
aws iam list-attached-role-policies --role-name my-role --query
"AttachedPolicies[].PolicyArn" --output text
```

Contoh output adalah sebagai berikut.

```
arn:aws:iam::111122223333:policy/my-policy
```

3. Tetapkan variabel untuk menyimpan Nama Sumber Daya Amazon (ARN) kebijakan yang ingin Anda gunakan. Ganti *my-policy* dengan nama kebijakan yang ingin Anda konfirmasi izin.

```
export policy_arn=arn:aws: iam::111122223333:policy/my-policy
```

4. Lihat versi default kebijakan.

```
aws iam get-policy --policy-arn $policy_arn
```

Contoh output adalah sebagai berikut.

```
{
  "Policy": {
    "PolicyName": "my-policy",
    "PolicyId": "EXAMPLEBIOWGLDEXAMPLE",
    "Arn": "arn:aws: iam::111122223333:policy/my-policy",
    "Path": "/",
    "DefaultVersionId": "v1",
    [...]
  }
}
```

5. Lihat konten kebijakan untuk memastikan bahwa kebijakan tersebut menyertakan semua izin yang dibutuhkan Pod Anda. Jika perlu, ganti **1** dalam perintah berikut dengan versi yang dikembalikan pada output sebelumnya.

```
aws iam get-policy-version --policy-arn $policy_arn --version-id v1
```

Contoh output adalah sebagai berikut.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:GetObject",
      "Resource": "arn:aws: s3::my-pod-secrets-bucket"
    }
  ]
}
```

Jika Anda membuat kebijakan contoh di langkah sebelumnya, maka output Anda sama. Jika Anda membuat kebijakan yang berbeda, maka *example* kontennya berbeda.

6. Konfirmasikan bahwa akun layanan Kubernetes dianotasi dengan peran tersebut.

```
kubectl describe serviceaccount my-service-account -n default
```

Contoh output adalah sebagai berikut.

```
Name:                my-service-account
Namespace:           default
Annotations:         eks.amazonaws.com/role-arn: arn:aws: iam::111122223333:role/my-
                    role
Image pull secrets:   <none>
Mountable secrets:    my-service-account-token-qqjfl
Tokens:              my-service-account-token-qqjfl
[...]
```

Langkah selanjutnya

- [the section called “Tetapkan ke Pod”](#)

Konfigurasi Pod untuk menggunakan akun layanan Kubernetes

Jika sebuah Pod perlu mengakses AWS layanan, maka Anda harus mengonfigurasinya untuk menggunakan akun layanan Kubernetes. Akun layanan harus dikaitkan dengan peran AWS Identity and Access Management (IAM) yang memiliki izin untuk mengakses layanan. AWS

- Sebuah kluster yang sudah ada. Jika Anda tidak memilikinya, Anda dapat membuatnya menggunakan salah satu panduan di [Memulai](#).
- Penyedia IAM OpenID Connect (OIDC) yang sudah ada untuk kluster Anda. Untuk mengetahui apakah Anda sudah memilikinya atau cara membuatnya, lihat [the section called “Penyedia IAM OIDC”](#).
- Akun layanan Kubernetes yang sudah ada yang terkait dengan peran IAM. Akun layanan harus dianotasi dengan Nama Sumber Daya Amazon (ARN) dari peran IAM. Peran tersebut harus memiliki kebijakan IAM terkait yang berisi izin yang Anda inginkan untuk Pod Anda untuk

menggunakan AWS layanan. Untuk informasi selengkapnya tentang cara membuat akun dan peran layanan, dan mengonfigurasinya, lihat [the section called “Tetapkan peran IAM”](#).

- Versi 2.12.3 atau yang lebih baru atau versi 1.27.160 atau yang lebih baru dari AWS Command Line Interface (AWS CLI) diinstal dan dikonfigurasi pada perangkat Anda atau AWS CloudShell. Untuk memeriksa versi Anda saat ini, gunakan `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Package manager seperti `yum` atau `apt-get`, atau Homebrew untuk macOS seringkali merupakan beberapa versi di belakang versi terbaru CLI. AWS Untuk menginstal versi terbaru, lihat [Menginstal](#) dan [Konfigurasi cepat dengan aws configure](#) di Panduan Pengguna Antarmuka Baris AWS Perintah. Versi AWS CLI yang diinstal AWS CloudShell mungkin juga beberapa versi di belakang versi terbaru. Untuk memperbaruinya, lihat [Menginstal AWS CLI ke direktori home Anda](#) di AWS CloudShell Panduan Pengguna.
 - Alat baris `kubectl` perintah diinstal pada perangkat Anda atau AWS CloudShell. Versinya bisa sama dengan atau hingga satu versi minor lebih awal atau lebih lambat dari versi Kubernetes dari kluster Anda. Misalnya, jika versi cluster Anda 1.29, Anda dapat menggunakan `kubectl` versi 1.28, 1.29, atau 1.30 dengan itu. Untuk menginstal atau memutakhirkan `kubectl`, lihat [the section called “Siapkan kubectl dan eksctl”](#).
 - `kubectl config` file yang ada yang berisi konfigurasi cluster Anda. Untuk membuat `kubectl config` file, lihat [the section called “Akses kluster dengan kubectl”](#).
1. Gunakan perintah berikut untuk membuat manifes deployment yang dapat digunakan untuk meng-deploy Pod untuk mengonfirmasi konfigurasi. Ganti *example values* dengan nilai-nilai milik Anda sendiri.

```
cat >my-deployment.yaml <<EOF
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      serviceAccountName: my-service-account
      containers:
```

```
- name: my-app
  image: public.ecr.aws/nginx/nginx:X.XX
EOF
```

2. Terapkan manifes ke cluster Anda.

```
kubectl apply -f my-deployment.yaml
```

3. Konfirmasikan bahwa variabel lingkungan yang diperlukan ada untuk Pod Anda.

a. Lihat Pod yang di-deploy dengan deployment pada langkah sebelumnya.

```
kubectl get pods | grep my-app
```

Contoh output adalah sebagai berikut.

```
my-app-6f4dfff6cb-76cv9    1/1      Running    0          3m28s
```

b. Lihat ARN dari peran IAM yang digunakan Pod.

```
kubectl describe pod my-app-6f4dfff6cb-76cv9 | grep AWS_ROLE_ARN:
```

Contoh output adalah sebagai berikut.

```
AWS_ROLE_ARN: arn:aws:iam::111122223333:role/my-role
```

Peran ARN harus cocok dengan ARN peran yang Anda anotasi dengan akun layanan yang ada. Untuk selengkapnya tentang membuat anotasi akun layanan, lihat [the section called “Tetapkan peran IAM”](#)

c. Konfirmasikan bahwa Pod memiliki mount file token identitas web.

```
kubectl describe pod my-app-6f4dfff6cb-76cv9 | grep AWS_WEB_IDENTITY_TOKEN_FILE:
```

Contoh output adalah sebagai berikut.

```
AWS_WEB_IDENTITY_TOKEN_FILE: /var/run/secrets/eks.amazonaws.com/serviceaccount/
token
```

kubelet meminta dan menyimpan token atas nama Pod. Secara default, token akan kubelet menyegarkan jika token lebih tua dari 80 persen dari total waktu untuk hidup atau lebih dari 24 jam. Anda dapat mengubah durasi kedaluwarsa untuk setiap akun selain akun layanan default dengan menggunakan pengaturan dalam spesifikasi Pod Anda. Untuk informasi selengkapnya, lihat [Service Account Token Volume Projection](#) dalam dokumentasi Kubernetes.

[Amazon EKS Pod Identity Webhook](#) di cluster mengawasi Pod yang menggunakan akun layanan dengan anotasi berikut:

```
eks.amazonaws.com/role-arn: arn:aws:iam::111122223333:role/my-role
```

Webhook menerapkan variabel lingkungan sebelumnya ke Pod tersebut. Cluster Anda tidak perlu menggunakan webhook untuk mengonfigurasi variabel lingkungan dan pemasangan file token. Anda dapat mengonfigurasi Pod secara manual untuk memiliki variabel lingkungan ini. [Versi AWS SDK yang didukung](#) mencari variabel lingkungan ini terlebih dahulu di penyedia rantai kredensial. Kredensial peran digunakan untuk Pod yang memenuhi kriteria ini.

4. Konfirmasikan bahwa Pod Anda dapat berinteraksi dengan AWS layanan menggunakan izin yang Anda tetapkan dalam kebijakan IAM yang dilampirkan pada peran Anda.

 Note

Ketika sebuah Pod menggunakan AWS kredensial dari peran IAM yang terkait dengan akun layanan, AWS CLI atau lainnya SDKs dalam kontainer untuk Pod tersebut menggunakan kredensial yang disediakan oleh peran tersebut. Jika Anda tidak membatasi akses ke kredensial yang diberikan ke [peran IAM node Amazon EKS](#), Pod masih memiliki akses ke kredensial-kredensialnya. Untuk informasi selengkapnya, lihat [Membatasi akses ke profil instance yang ditetapkan ke node pekerja](#).

Jika Pod Anda tidak dapat berinteraksi dengan layanan seperti yang Anda harapkan, selesaikan langkah-langkah berikut untuk mengonfirmasi bahwa semuanya telah dikonfigurasi dengan benar.

- a. Konfirmasikan bahwa Pod Anda menggunakan versi AWS SDK yang mendukung asumsi peran IAM melalui file token identitas web OpenID Connect. Untuk informasi selengkapnya, lihat [the section called “Didukung SDKs”](#).

- b. Konfirmasikan bahwa penyebaran menggunakan akun layanan.

```
kubectl describe deployment my-app | grep "Service Account"
```

Contoh output adalah sebagai berikut.

```
Service Account: my-service-account
```

- c. Jika Pod Anda masih tidak dapat mengakses layanan, tinjau [langkah-langkah](#) yang dijelaskan dalam [Menetapkan peran IAM ke akun layanan Kubernetes untuk mengonfirmasi bahwa akun peran dan layanan](#) Anda telah dikonfigurasi dengan benar.

Konfigurasi titik akhir Layanan Token AWS Keamanan untuk akun layanan

Jika Anda menggunakan akun layanan Kubernetes dengan [peran IAM untuk akun layanan](#), maka Anda dapat mengonfigurasi tipe titik akhir Layanan Token AWS Keamanan yang digunakan oleh akun layanan jika versi cluster dan platform Anda sama atau lebih lambat dari yang tercantum dalam tabel berikut. Jika Kubernetes atau versi platform Anda lebih awal dari yang tercantum dalam tabel, maka akun layanan Anda hanya dapat menggunakan endpoint global.

Versi Kubernetes	Versi platform	Jenis titik akhir default
1.31	eks.4	Regional
1.30	eks.2	Regional
1.29	eks.1	Regional
1.28	eks.1	Regional
1.27	eks.1	Regional
1.26	eks.1	Regional

AWS merekomendasikan menggunakan titik akhir AWS STS regional alih-alih titik akhir global. Ini mengurangi latensi, menyediakan redundansi bawaan, dan meningkatkan validitas token sesi. AWS Security Token Service harus aktif di AWS Region dimana Pod berjalan. Selain itu, aplikasi Anda harus memiliki redundansi bawaan untuk AWS Wilayah yang berbeda jika terjadi kegagalan

layanan di AWS Wilayah. Untuk informasi selengkapnya, lihat [Mengelola AWS STS di AWS Wilayah](#) di Panduan Pengguna IAM.

- Sebuah klaster yang sudah ada. Jika Anda tidak memilikinya, Anda dapat membuatnya menggunakan salah satu panduan di [Memulai](#).
- Penyedia IAM OIDC yang sudah ada untuk klaster Anda. Untuk informasi selengkapnya, lihat [the section called “Penyedia IAM OIDC”](#).
- Akun layanan Kubernetes yang sudah ada yang dikonfigurasi untuk digunakan dengan fitur [Amazon EKS IAM untuk](#) akun layanan.

[Contoh berikut semuanya menggunakan akun layanan Kubernetes aws-node yang digunakan oleh plugin Amazon VPC CNI](#). Anda dapat menggantinya *example values* dengan akun layanan, Pod, ruang nama, dan sumber daya lainnya.

1. Pilih Pod yang menggunakan akun layanan yang ingin Anda ubah endpoint. Tentukan AWS Region mana Pod berjalan. Ganti *aws-node-6mfgv* dengan nama Pod Anda dan *kube-system* dengan namespace Pod Anda.

```
kubectl describe pod aws-node-6mfgv -n kube-system |grep Node:
```

Contoh output adalah sebagai berikut.

```
ip-192-168-79-166.us-west-2/192.168.79.166
```

Pada output sebelumnya, Pod berjalan pada sebuah node di Region us-west-2 AWS .

2. Tentukan tipe endpoint yang digunakan oleh akun layanan Pod.

```
kubectl describe pod aws-node-6mfgv -n kube-system |grep AWS_STS_REGIONAL_ENDPOINTS
```

Contoh output adalah sebagai berikut.

```
AWS_STS_REGIONAL_ENDPOINTS: regional
```

Jika titik akhir saat ini bersifat global, `global` maka dikembalikan dalam output. Jika tidak ada output yang dikembalikan, maka tipe endpoint default sedang digunakan dan belum diganti.

3. Jika versi cluster atau platform Anda sama atau lebih lambat dari yang tercantum dalam tabel, maka Anda dapat mengubah jenis titik akhir yang digunakan oleh akun layanan Anda dari tipe default ke tipe yang berbeda dengan salah satu perintah berikut. Ganti `aws-node` dengan nama akun layanan Anda dan `kube-system` dengan namespace untuk akun layanan Anda.

- Jika tipe titik akhir default atau saat ini bersifat global dan Anda ingin mengubahnya menjadi regional:

```
kubectl annotate serviceaccount -n kube-system aws-node eks.amazonaws.com/sts-regional-endpoints=true
```

Jika Anda menggunakan [peran IAM untuk akun layanan guna](#) menghasilkan S3 yang telah ditandatangani sebelumnya URLs dalam aplikasi yang berjalan di container Pods, format URL untuk titik akhir regional mirip dengan contoh berikut:

```
https://bucket.s3.us-west-2.amazonaws.com/path?...&X-Amz-Credential=your-access-key-id/date/us-west-2/s3/aws4_request&...
```

- Jika tipe titik akhir default atau saat ini bersifat regional dan Anda ingin mengubahnya menjadi global:

```
kubectl annotate serviceaccount -n kube-system aws-node eks.amazonaws.com/sts-regional-endpoints=false
```

Jika aplikasi Anda secara eksplisit membuat permintaan ke titik akhir global AWS STS dan Anda tidak mengganti perilaku default menggunakan titik akhir regional di kluster Amazon EKS, permintaan akan gagal dengan kesalahan. Untuk informasi selengkapnya, lihat [the section called “Kontainer pod menerima kesalahan berikut: An error occurred \(SignatureDoesNotMatch\) when calling the GetCallerIdentity operation: Credential should be scoped to a valid region”](#).

Jika Anda menggunakan [peran IAM untuk akun layanan guna](#) menghasilkan S3 yang telah ditandatangani sebelumnya URLs dalam aplikasi yang berjalan di container Pods, format URL untuk titik akhir global mirip dengan contoh berikut:

```
https://bucket.s3.amazonaws.com/path?...&X-Amz-Credential=your-access-key-id/date/us-west-2/s3/aws4_request&...
```

Jika Anda memiliki otomatisasi yang mengharap URL yang telah ditandatangani sebelumnya dalam format tertentu atau jika aplikasi atau dependensi hilir yang menggunakan pra-tanda tangan URLs memiliki harapan untuk AWS Wilayah yang ditargetkan, maka buat perubahan yang diperlukan untuk menggunakan titik akhir STS yang sesuai. AWS

4. Hapus dan buat ulang Pod yang ada yang terkait dengan akun layanan untuk menerapkan variabel lingkungan kredensial. Web hook yang bermutasi tidak menerapkannya pada Pod yang sudah berjalan. Anda dapat mengganti `Pods`, `kube-system`, dan `-l k8s-app=aws-node` dengan informasi untuk Pod yang Anda tetapkan anotasi.

```
kubectl delete Pods -n kube-system -l k8s-app=aws-node
```

5. Konfirmasikan bahwa semua Pod dimulai ulang.

```
kubectl get Pods -n kube-system -l k8s-app=aws-node
```

6. Lihat variabel lingkungan untuk salah satu Pod. Verifikasi bahwa `AWS_STS_REGIONAL_ENDPOINTS` nilainya adalah apa yang Anda atur pada langkah sebelumnya.

```
kubectl describe pod aws-node-kzbtr -n kube-system |grep AWS_STS_REGIONAL_ENDPOINTS
```

Contoh output adalah sebagai berikut.

```
AWS_STS_REGIONAL_ENDPOINTS=regional
```

Otentikasi ke akun lain dengan IRSA

Anda dapat mengonfigurasi izin IAM lintas akun baik dengan membuat penyedia identitas dari kluster akun lain atau dengan menggunakan operasi berantai. `AssumeRole` Dalam contoh berikut, Akun A memiliki kluster Amazon EKS yang mendukung peran IAM untuk akun layanan. Pod yang berjalan di kluster tersebut harus mengasumsikan izin IAM dari Akun B.

Example Buat penyedia identitas dari kluster akun lain

Example

Dalam contoh ini, Akun A menyediakan Akun B dengan URL penerbit OpenID Connect (OIDC) dari kluster mereka. Akun B mengikuti petunjuk dalam [Membuat penyedia IAM OIDC untuk kluster Anda](#) dan [the section called “Tetapkan peran IAM”](#) menggunakan URL penerbit OIDC dari kluster Akun

A. Kemudian, administrator kluster memberi anotasi pada akun layanan di kluster Akun A untuk menggunakan peran dari Akun B (**444455556666**).

```
apiVersion: v1
kind: ServiceAccount
metadata:
  annotations:
    eks.amazonaws.com/role-arn: arn:aws: iam::444455556666:role/account-b-role
```

Example Gunakan operasi berantai **AssumeRole**

Example

Dalam contoh ini, Akun B membuat kebijakan IAM dengan izin untuk diberikan ke Pod di kluster Akun A. Akun B (**444455556666**) melampirkan kebijakan tersebut ke peran IAM dengan hubungan kepercayaan yang memungkinkan AssumeRole izin ke Akun A (**111122223333**).

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws: iam::111122223333:root"
      },
      "Action": "sts:AssumeRole",
      "Condition": {}
    }
  ]
}
```

Akun A membuat peran dengan kebijakan kepercayaan yang mendapatkan kredensi dari penyedia identitas yang dibuat dengan alamat penerbit OIDC kluster.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws: iam::111122223333:oidc-provider/oidc.eks.region-
code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
```

```
    },
    "Action": "sts:AssumeRoleWithWebIdentity"
  }
]
}
```

Akun A melampirkan kebijakan ke peran tersebut untuk mengambil peran yang dibuat oleh Akun B dengan izin berikut.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "arn:aws: iam::444455556666:role/account-b-role"
    }
  ]
}
```

Kode aplikasi untuk Pod untuk mengambil peran Akun B menggunakan dua profil: `account_b_role` dan `account_a_role`. Profil `account_b_role` menggunakan profil `account_a_role` sebagai sumbernya. Untuk AWS CLI, `~/.aws/config` file ini mirip dengan yang berikut ini.

```
[profile account_b_role]
source_profile = account_a_role
role_arn=arn:aws: iam::444455556666:role/account-b-role

[profile account_a_role]
web_identity_token_file = /var/run/secrets/eks.amazonaws.com/serviceaccount/token
role_arn=arn:aws: iam::111122223333:role/account-a-role
```

Untuk menentukan profil berantai untuk lainnya AWS SDKs, lihat dokumentasi SDK yang Anda gunakan. Untuk informasi selengkapnya, lihat [Alat untuk Dibangun AWS](#).

Gunakan IRSA dengan SDK AWS

Menggunakan kredensialnya

Untuk menggunakan kredensi dari peran IAM untuk akun layanan (IRSA), kode Anda dapat menggunakan SDK apa pun untuk membuat klien untuk AWS layanan dengan AWS SDK, dan secara default SDK mencari di rantai lokasi untuk kredensi Identity and Access Management AWS

yang akan digunakan. Peran IAM untuk kredensial akun layanan akan digunakan jika Anda tidak menentukan penyedia kredensial saat membuat klien atau menginisialisasi SDK.

Ini berfungsi karena peran IAM untuk akun layanan telah ditambahkan sebagai langkah dalam rantai kredensial default. Jika beban kerja Anda saat ini menggunakan kredensial yang sebelumnya dalam rantai kredensial, kredensial tersebut akan terus digunakan bahkan jika Anda mengonfigurasi peran IAM untuk akun layanan untuk beban kerja yang sama.

SDK secara otomatis menukar token OIDC akun layanan untuk kredensial sementara dari AWS Security Token Service dengan menggunakan tindakan tersebut. `AssumeRoleWithWebIdentity` Amazon EKS dan tindakan SDK ini terus memutar kredensial sementara dengan memperbaruinya sebelum kedaluwarsa.

Saat menggunakan [peran IAM untuk akun layanan](#), container di Pod Anda harus menggunakan versi AWS SDK yang mendukung asumsi peran IAM melalui file token identitas web OpenID Connect. Pastikan Anda menggunakan versi berikut, atau yang lebih baru, untuk AWS SDK Anda:

- Java (Versi 2) - [2.10.11](#)
- Jawa — [1.12.782](#)
- AWS [SDK for Go v1 - 1.23.13](#)
- AWS SDK for Go v2 - Semua versi mendukung
- [Python \(Boto3\) - 1.9.220](#)
- [Python \(botocore\) - 1.12.200](#)
- AWS [CLI — 1.16.232](#)
- [Node — 2.525.0 dan 3.27.0](#)
- [Ruby — 3.58.0](#)
- [C++ — 1.7.174](#)
- .NET — [3.3.659.1](#) — Anda juga harus menyertakan `AWSSDK.SecurityToken`
- PHP — [3.110.7](#)

Banyak add-on Kubernetes yang populer, seperti [Cluster Autoscaler](#), [lalu lintas internet Route dengan Load AWS Balancer Controller](#), dan plugin [Amazon VPC](#) CNI untuk Kubernetes mendukung peran IAM untuk akun layanan.

Untuk memastikan bahwa Anda menggunakan SDK yang didukung, ikuti petunjuk penginstalan untuk SDK pilihan Anda di [Tools to Build on AWS](#) saat Anda membuat container.

Pertimbangan

Java

Saat menggunakan Java, Anda harus menyertakan sts modul pada classpath. Untuk informasi selengkapnya, lihat [WebIdentityTokenFileCredentialsProvider](#) di dokumen Java SDK.

Ambil kunci penandatanganan untuk memvalidasi token OIDC

Kubernetes mengeluarkan a ProjectedServiceAccountToken ke setiap Akun Layanan Kubernetes. Token ini adalah token OIDC, yang selanjutnya merupakan jenis token web JSON (JWT). Amazon EKS menghosting endpoint OIDC publik untuk setiap cluster yang berisi kunci penandatanganan untuk token sehingga sistem eksternal dapat memvalidasinya.

Untuk memvalidasi ProjectedServiceAccountToken, Anda perlu mengambil kunci penandatanganan publik OIDC, juga disebut JSON Web Key Set (JWKS). Gunakan kunci ini dalam aplikasi Anda untuk memvalidasi token. Misalnya, Anda dapat menggunakan pustaka [PyJWT Python](#) untuk memvalidasi token menggunakan kunci ini. Untuk informasi lebih lanjut tentang ProjectedServiceAccountToken, lihat [the section called “Informasi latar belakang IAM, Kubernetes, dan OpenID Connect \(OIDC\)”](#).

Prasyarat

- Penyedia OpenID Connect (OIDC) AWS Identity and Access Management (IAM) yang sudah ada untuk kluster Anda. Untuk menentukan apakah Anda sudah memiliki satu, atau harus membuat satu, lihat [the section called “Penyedia IAM OIDC”](#).
- AWS CLI — Alat baris perintah untuk bekerja dengan AWS layanan, termasuk Amazon EKS. Untuk informasi selengkapnya, lihat [Menginstal](#) di Panduan Pengguna Antarmuka Baris AWS Perintah. Setelah menginstal AWS CLI, kami sarankan Anda juga mengkonfigurasinya. Untuk informasi selengkapnya, lihat [Konfigurasi cepat dengan aws configure](#) di Panduan Pengguna Antarmuka Baris AWS Perintah.

Prosedur

1. Ambil URL OIDC untuk kluster Amazon EKS Anda menggunakan CLI. AWS

```
$ aws eks describe-cluster --name my-cluster --query 'cluster.identity.oidc.issuer'  
"https://oidc.eks.us-west-2.amazonaws.com/id/8EBDXXX00BAE"
```

2. Ambil kunci penandatanganan publik menggunakan curl, atau alat serupa. Hasilnya adalah [JSON Web Key Set \(JWKS\)](#).

 Important

Amazon EKS membatasi panggilan ke titik akhir OIDC. Anda harus men-cache kunci penandatanganan publik. Hormati `cache-control` header yang termasuk dalam respons.

 Important

Amazon EKS memutar kunci penandatanganan OIDC setiap tujuh hari.

```
$ curl https://oidc.eks.us-west-2.amazonaws.com/id/8EBDXXXX00BAE/keys
{"keys":
[{"kty":"RSA","kid":"2284XXXX4a40","use":"sig","alg":"RS256","n":"wk1bXXXXMVfQ","e":"AQAB"}]}
```

Pelajari cara EKS Pod Identity memberikan akses Pod ke layanan AWS

Aplikasi dalam kontainer Pod dapat menggunakan AWS SDK atau AWS CLI untuk membuat permintaan API AWS ke layanan AWS menggunakan izin Identity and Access Management (IAM). Aplikasi harus menandatangani permintaan AWS API mereka dengan AWS kredensialnya.

EKS Pod Identities menyediakan kemampuan untuk mengelola kredensial untuk aplikasi Anda, mirip dengan cara profil EC2 instans Amazon memberikan kredensi ke instans Amazon. EC2 Alih-alih membuat dan mendistribusikan AWS kredensialmu ke container atau menggunakan peran EC2 instans Amazon, kamu mengaitkan peran IAM dengan akun layanan Kubernetes dan mengonfigurasi Pod kamu untuk menggunakan akun layanan.

Setiap asosiasi EKS Pod Identity memetakan peran ke akun layanan di namespace di cluster yang ditentukan. Jika Anda memiliki aplikasi yang sama di beberapa cluster, Anda dapat membuat asosiasi identik di setiap cluster tanpa mengubah kebijakan kepercayaan peran.

Jika sebuah pod menggunakan akun layanan yang memiliki asosiasi, Amazon EKS menetapkan variabel lingkungan dalam kontainer pod. Variabel lingkungan mengonfigurasi AWS SDKs, termasuk AWS CLI, untuk menggunakan kredensial Identitas Pod EKS.

Manfaat Identitas EKS Pod

EKS Pod Identities memberikan manfaat sebagai berikut:

- Keistimewaan paling kecil — Anda dapat memberikan cakupan izin IAM ke akun layanan, dan hanya Pod yang menggunakan akun layanan tersebut yang memiliki akses ke izin tersebut. Fitur ini juga mengurangi kebutuhan akan solusi dari pihak ketiga seperti kiam atau kube2iam.
- Isolasi kredensial — Ketika akses ke [Amazon EC2 Instance Metadata Service \(IMDS\)](#) dibatasi, container Pod hanya dapat mengambil kredensial untuk peran IAM yang terkait dengan akun layanan yang digunakan container. Container tidak pernah memiliki akses ke kredensi yang digunakan oleh kontainer lain di Pod lain. Jika IMDS tidak dibatasi, kontainer Pod juga memiliki akses ke [peran IAM node Amazon EKS](#) dan kontainer mungkin dapat memperoleh akses ke kredensial-kredensi peran IAM dari Pod lain pada node yang sama. Untuk informasi selengkapnya, lihat [Membatasi akses ke profil instance yang ditetapkan ke node pekerja](#).

Note

Pod yang dikonfigurasi `hostNetwork: true` akan selalu memiliki akses IMDS, tetapi CLI AWS SDKs dan akan menggunakan kredensial Identitas Pod saat diaktifkan.

- Auditabilitas — Akses dan pencatatan peristiwa tersedia AWS CloudTrail untuk membantu memfasilitasi audit retrospektif.

Important

Container bukanlah batas keamanan, dan penggunaan Pod Identity tidak mengubah ini. Pod yang ditetapkan ke node yang sama akan berbagi kernel dan kemungkinan sumber daya lainnya tergantung pada konfigurasi Pod Anda. Sementara Pod yang berjalan pada node terpisah akan diisolasi pada lapisan komputasi, ada aplikasi node yang memiliki izin tambahan di API Kubernetes di luar cakupan instance individual. Beberapa contohnya adalah kubelet, kube-proxy, driver penyimpanan CSI, atau aplikasi Kubernetes Anda sendiri.

EKS Pod Identity adalah metode yang lebih sederhana daripada [the section called “Kredensial dengan IRSA”](#), karena metode ini tidak menggunakan penyedia identitas OIDC. EKS Pod Identity memiliki penyempurnaan sebagai berikut:

- **Operasi independen** — Di banyak organisasi, membuat penyedia identitas OIDC adalah tanggung jawab tim yang berbeda daripada mengelola kluster Kubernetes. EKS Pod Identity memiliki pemisahan tugas yang bersih, di mana semua konfigurasi asosiasi Identitas Pod EKS dilakukan di Amazon EKS dan semua konfigurasi izin IAM dilakukan di IAM.
- **Reusability** — EKS Pod Identity menggunakan satu prinsipal IAM, bukan prinsipal terpisah untuk setiap kluster yang digunakan peran IAM untuk akun layanan. Administrator IAM Anda menambahkan prinsipal berikut ke kebijakan kepercayaan dari peran apa pun agar dapat digunakan oleh EKS Pod Identities.

```
"Principal": {  
  "Service": "pods.eks.amazonaws.com"  
}
```

- **Skalabilitas** — Setiap set kredensial sementara diasumsikan oleh layanan EKS Auth di EKS Pod Identity, bukan setiap AWS SDK yang Anda jalankan di setiap pod. Kemudian, Amazon EKS Pod Identity Agent yang berjalan di setiap node mengeluarkan kredensialnya ke SDKs. Dengan demikian beban dikurangi menjadi satu kali untuk setiap node dan tidak diduplikasi di setiap pod. Untuk detail lebih lanjut tentang prosesnya, lihat [the section called “Cara kerjanya”](#).

Untuk informasi lebih lanjut untuk membandingkan dua alternatif, lihat [the section called “Akses beban kerja ke AWS”](#).

Ikhtisar pengaturan EKS Pod Identities

Aktifkan Identitas Pod EKS dengan menyelesaikan prosedur berikut:

1. [the section called “Mengatur Agen”](#)— Anda hanya menyelesaikan prosedur ini sekali untuk setiap cluster. Anda tidak perlu menyelesaikan langkah ini jika Mode Otomatis EKS diaktifkan di cluster Anda.
2. [the section called “Tetapkan peran IAM”](#)— Selesaikan prosedur ini untuk setiap set izin unik yang Anda inginkan untuk dimiliki aplikasi.
3. [the section called “Akun layanan Pod”](#)— Selesaikan prosedur ini untuk setiap Pod yang membutuhkan akses ke AWS layanan.

4. [the section called “Didukung SDKs”](#)— Konfirmasikan bahwa beban kerja menggunakan AWS SDK versi yang didukung dan bahwa beban kerja menggunakan rantai kredensial default.

Batas

- Hingga 5.000 asosiasi Identitas Pod EKS per cluster untuk memetakan peran IAM ke akun layanan Kubernetes didukung.

Pertimbangan

- Asosiasi Peran IAM: Setiap akun layanan Kubernetes dalam kluster dapat dikaitkan dengan satu peran IAM dari akun yang sama AWS dengan kluster. Untuk mengubah peran, edit asosiasi EKS Pod Identity. Untuk akses lintas akun, delegasikan akses ke peran menggunakan peran IAM. Untuk mempelajari lebih lanjut, lihat [Mendelegasikan akses di seluruh AWS akun menggunakan peran IAM](#) di Panduan Pengguna IAM.
- Agen Identitas Pod EKS: Agen Identitas Pod wajib menggunakan EKS Pod Identity. Agen berjalan sebagai Kubernetes DaemonSet pada node cluster, memberikan kredensi hanya untuk pod pada node yang sama. Ini menggunakan nodehostNetwork, menempati port 80 dan 2703 pada alamat link-lokal (169.254.170.23 untuk IPv4, [fd00:ec2::23] untuk IPv6). Jika IPv6 dinonaktifkan di kluster Anda, nonaktifkan IPv6 untuk Agen Identitas Pod. Untuk mempelajari lebih lanjut, lihat [Menonaktifkan IPv6 di Agen Identitas Pod EKS](#).
- Konsistensi Akhir: Asosiasi Identitas Pod EKS pada akhirnya konsisten, dengan potensi penundaan beberapa detik setelah panggilan API. Hindari membuat atau memperbarui asosiasi di jalur kode ketersediaan tinggi yang kritis. Sebagai gantinya, lakukan tindakan ini dalam rutinitas inisialisasi atau pengaturan yang terpisah dan lebih jarang. Untuk mempelajari selengkapnya, lihat [Grup Keamanan Per Pod](#) di Panduan Praktik Terbaik EKS.
- Pertimbangan Grup Proxy dan Keamanan: Untuk pod yang menggunakan proxy, tambahkan 169.254.170.23 (IPv4) dan [fd00:ec2::23] (IPv6) ke variabel no_proxy/NO_PROXY lingkungan untuk mencegah permintaan gagal ke Agen Identitas Pod EKS. Jika menggunakan Grup Keamanan untuk Pod dengan AWS VPC CNI, atur ENABLE_POD_ENI flag ke 'true' dan flag ke 'standard'. POD_SECURITY_GROUP_ENFORCING_MODE Untuk mempelajari lebih lanjut, lihat [Menetapkan grup keamanan ke Pod individual](#).

EKS Pod Identity versi cluster

Untuk menggunakan EKS Pod Identity, cluster harus memiliki versi platform yang sama atau lebih lambat dari versi yang tercantum dalam tabel berikut, atau versi Kubernetes yang lebih lambat dari versi yang tercantum dalam tabel. Untuk menemukan versi yang disarankan dari Amazon EKS Pod Identity Agent untuk versi Kubernetes, lihat. [the section called “Verifikasi kompatibilitas”](#)

Versi Kubernetes	Versi platform
Versi Kubernetes tidak terdaftar	Semua versi platform mendukung
1.28	eks.4
1.27	eks.8
1.26	eks.9

Pembatasan Identitas Pod EKS

Identitas Pod EKS tersedia pada hal-hal berikut:

- Amazon EKS versi cluster tercantum dalam topik sebelumnya [the section called “EKS Pod Identity versi cluster”](#).
- Node pekerja di cluster yang merupakan EC2 instance Linux Amazon.

EKS Pod Identities tidak tersedia di berikut ini:

- AWS Outposts.
- Amazon EKS Anywhere.
- Cluster Kubernetes yang Anda buat dan jalankan di Amazon. EC2 Komponen EKS Pod Identity hanya tersedia di Amazon EKS.

Anda tidak dapat menggunakan EKS Pod Identities dengan:

- Pod yang berjalan di mana saja kecuali EC2 instance Linux Amazon. Pod Linux dan Windows yang berjalan di AWS Fargate (Fargate) tidak didukung. Pod yang berjalan di EC2 instans Windows Amazon tidak didukung.

Pahami cara kerja EKS Pod Identity

Asosiasi Amazon EKS Pod Identity menyediakan kemampuan untuk mengelola kredensial untuk aplikasi Anda, mirip dengan cara profil EC2 instans Amazon memberikan kredensial ke instans Amazon. EC2

Amazon EKS Pod Identity memberikan kredensi ke beban kerja Anda dengan API Auth EKS tambahan dan pod agen yang berjalan di setiap node.

Di add-on Anda, seperti add-on Amazon EKS dan pengontrol yang dikelola sendiri, operator, dan add-on lainnya, penulis perlu memperbarui perangkat lunak mereka untuk menggunakan yang terbaru. AWS SDKs Untuk daftar kompatibilitas antara EKS Pod Identity dan add-on yang diproduksi oleh Amazon EKS, lihat bagian [the section called “Pembatasan Identitas Pod EKS”](#) sebelumnya.

Menggunakan Identitas Pod EKS dalam kode Anda

Dalam kode Anda, Anda dapat menggunakan AWS SDKs untuk mengakses AWS layanan. Anda menulis kode untuk membuat klien untuk AWS layanan dengan SDK, dan secara default SDK mencari di rantai lokasi untuk menggunakan kredensial AWS Identity and Access Management. Setelah kredensi yang valid ditemukan, pencarian dihentikan. Untuk informasi selengkapnya tentang lokasi default yang digunakan, lihat [rantai penyedia kredensial](#) di Panduan Referensi Alat AWS SDKs dan Alat.

EKS Pod Identities telah ditambahkan ke penyedia kredensi Container yang dicari dalam satu langkah dalam rantai kredensi default. Jika beban kerja Anda saat ini menggunakan kredensial yang sebelumnya ada dalam rantai kredensial, kredensial-kredensialnya akan terus digunakan meskipun Anda mengonfigurasi asosiasi Identitas Pod EKS untuk beban kerja yang sama. Dengan cara ini Anda dapat dengan aman bermigrasi dari jenis kredensial lain dengan membuat asosiasi terlebih dahulu, sebelum menghapus kredensial lama.

Penyedia kredensial kontainer menyediakan kredensi sementara dari agen yang berjalan di setiap node. Di Amazon EKS, agennya adalah Agen Identitas Pod Amazon EKS dan di Amazon Elastic Container Service agennya adalah `amazon-ecs-agent`. Variabel lingkungan SDKs penggunaan untuk menemukan agen untuk terhubung ke.

Sebaliknya, peran IAM untuk akun layanan menyediakan token identitas web yang harus ditukar oleh AWS SDK dengan AWS Security Token Service. `AssumeRoleWithWebIdentity`

Bagaimana Agen Identitas EKS Pod bekerja dengan Pod

1. Saat Amazon EKS memulai pod baru yang menggunakan akun layanan dengan asosiasi Identitas Pod EKS, kluster akan menambahkan konten berikut ke manifes Pod:

```
env:
  - name: AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE
    value: "/var/run/secrets/pods.eks.amazonaws.com/serviceaccount/eks-pod-identity-token"
  - name: AWS_CONTAINER_CREDENTIALS_FULL_URI
    value: "http://169.254.170.23/v1/credentials"
volumeMounts:
  - mountPath: "/var/run/secrets/pods.eks.amazonaws.com/serviceaccount/"
    name: eks-pod-identity-token
volumes:
  - name: eks-pod-identity-token
    projected:
      defaultMode: 420
      sources:
        - serviceAccountToken:
            audience: pods.eks.amazonaws.com
            expirationSeconds: 86400 # 24 hours
            path: eks-pod-identity-token
```

2. Kubernetes memilih node mana yang akan menjalankan pod. Kemudian, Amazon EKS Pod Identity Agent pada node menggunakan [AssumeRoleForPodIdentity](#) action untuk mengambil kredensial sementara dari EKS Auth API.
3. Agen Identitas Pod EKS membuat kredensial ini tersedia untuk AWS SDKs yang Anda jalankan di dalam kontainer Anda.
4. Anda menggunakan SDK dalam aplikasi Anda tanpa menentukan penyedia kredensi untuk menggunakan rantai kredensi default. Atau, Anda menentukan penyedia kredensi kontainer. Untuk informasi selengkapnya tentang lokasi default yang digunakan, lihat [rantai penyedia kredensial](#) di Panduan Referensi Alat AWS SDKs dan Alat.
5. SDK menggunakan variabel lingkungan untuk terhubung ke Agen Identitas Pod EKS dan mengambil kredensialnya.

 Note

Jika beban kerja Anda saat ini menggunakan kredensial yang sebelumnya ada dalam rantai kredensial, kredensial-kredensialnya akan terus digunakan meskipun Anda mengonfigurasi asosiasi Identitas Pod EKS untuk beban kerja yang sama.

Siapkan Agen Identitas Amazon EKS Pod

Asosiasi Amazon EKS Pod Identity menyediakan kemampuan untuk mengelola kredensial untuk aplikasi Anda, mirip dengan cara profil EC2 instans Amazon memberikan kredensi ke instans Amazon. EC2

Amazon EKS Pod Identity memberikan kredensi ke beban kerja Anda dengan API Auth EKS tambahan dan pod agen yang berjalan di setiap node.

 Tip

Anda tidak perlu menginstal Agen Identitas Pod EKS pada Kluster Mode Otomatis EKS. Kemampuan ini dibangun ke dalam Mode Otomatis EKS.

Pertimbangan

- Secara default, Agen Identitas Pod EKS mendengarkan IPv4 dan IPv6 alamat pod untuk meminta kredensialnya. Agen menggunakan alamat IP loopback (localhost) 169.254.170.23 untuk IPv4 dan alamat IP localhost untuk. [fd00:ec2::23] IPv6
- Jika Anda menonaktifkan IPv6 alamat, atau mencegah alamat IPv6 IP localhost, agen tidak dapat memulai. Untuk memulai agen pada node yang tidak dapat digunakan IPv6, ikuti langkah-langkah [the section called “Nonaktifkan IPv6”](#) untuk menonaktifkan IPv6 konfigurasi.

Membuat Agen Identitas Amazon EKS Pod

Prasyarat agen

- Sebuah kluster Amazon EKS yang sudah ada. Untuk menyebarkan satu, lihat [Memulai](#). Versi cluster dan versi platform harus sama atau lebih lambat dari versi yang tercantum dalam [versi cluster EKS Pod Identity](#).

- Peran node memiliki izin bagi agen untuk melakukan `AssumeRoleForPodIdentity` tindakan di EKS Auth API. Anda dapat menggunakan [kebijakan AWS terkelola: Amazon EKSWorkerNodePolicy](#) atau menambahkan kebijakan khusus yang serupa dengan berikut ini:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "eks-auth:AssumeRoleForPodIdentity"
      ],
      "Resource": "*"
    }
  ]
}
```

Tindakan ini dapat dibatasi oleh tag untuk membatasi peran mana yang dapat diasumsikan oleh pod yang menggunakan agen.

- Node dapat menjangkau dan mengunduh gambar dari Amazon ECR. Gambar kontainer untuk add-on ada di pendaftar yang tercantum di Lihat [pendaftar gambar kontainer Amazon untuk add-on Amazon EKS](#).

Perhatikan bahwa Anda dapat mengubah lokasi gambar dan `imagePullSecrets` menyediakan add-on EKS di pengaturan konfigurasi Opsional di AWS Management Console, dan `--configuration-values` di AWS CLI.

- Node dapat mencapai Amazon EKS Auth API. Untuk cluster pribadi, `eks-auth` titik akhir di AWS PrivateLink diperlukan.

Agen penyiapan dengan AWS konsol

1. Buka [konsol Amazon EKS](#).
2. Di panel navigasi kiri, pilih Clusters, lalu pilih nama cluster yang ingin Anda konfigurasikan untuk add-on EKS Pod Identity Agent.
3. Pilih tab Add-ons.
4. Pilih Get more add-ons

5. Pilih kotak di kanan atas kotak add-on untuk EKS Pod Identity Agent dan kemudian pilih Berikutnya.
6. Pada halaman Konfigurasi pengaturan add-on yang dipilih, pilih versi apa pun di daftar dropdown Versi.
7. (Opsional) Perluas pengaturan konfigurasi opsional untuk memasukkan konfigurasi tambahan. Misalnya, Anda dapat memberikan lokasi gambar kontainer alternatif dan `ImagePullSecrets`. Skema JSON dengan kunci yang diterima ditampilkan dalam skema konfigurasi Add-on.

Masukkan tombol konfigurasi dan nilai dalam nilai Konfigurasi.

8. Pilih Berikutnya.
9. Konfirmasikan bahwa pod EKS Pod Identity Agent berjalan di kluster Anda.

```
kubectl get pods -n kube-system | grep 'eks-pod-identity-agent'
```

Contoh output adalah sebagai berikut.

eks-pod-identity-agent-gmqp7	1/1	Running
1 (24h ago) 24h		
eks-pod-identity-agent-prnsh	1/1	Running
1 (24h ago) 24h		

Anda sekarang dapat menggunakan asosiasi EKS Pod Identity di kluster Anda. Untuk informasi selengkapnya, lihat [the section called “Tetapkan peran IAM”](#).

Agen penyiapan dengan AWS CLI

1. Jalankan perintah AWS CLI berikut. Ganti `my-cluster` dengan nama kluster Anda.

```
aws eks create-addon --cluster-name my-cluster --addon-name eks-pod-identity-agent --addon-version v1.0.0-eksbuild.1
```

Note

Agen Identitas Pod EKS tidak menggunakan peran *service-account-role-arn* untuk IAM untuk akun layanan. Anda harus memberikan izin kepada Agen Identitas Pod EKS dalam peran node.

2. Konfirmasikan bahwa pod EKS Pod Identity Agent berjalan di kluster Anda.

```
kubectl get pods -n kube-system | grep 'eks-pod-identity-agent'
```

Contoh output adalah sebagai berikut.

eks-pod-identity-agent-gmqp7	1/1	Running
1 (24h ago) 24h		
eks-pod-identity-agent-prnsh	1/1	Running
1 (24h ago) 24h		

Anda sekarang dapat menggunakan asosiasi EKS Pod Identity di kluster Anda. Untuk informasi selengkapnya, lihat [the section called “Tetapkan peran IAM”](#).

Menetapkan peran IAM ke akun layanan Kubernetes

Topik ini membahas cara mengkonfigurasi akun layanan Kubernetes untuk mengambil peran AWS Identity and Access Management (IAM) dengan EKS Pod Identity. Pod apa pun yang dikonfigurasi untuk menggunakan akun layanan kemudian dapat mengakses AWS layanan apa pun yang perannya memiliki izin untuk diakses.

Untuk membuat asosiasi EKS Pod Identity, hanya ada satu langkah; Anda membuat asosiasi di EKS melalui AWS Management Console, AWS CLI, AWS SDKs, AWS CloudFormation dan alat-alat lainnya. Tidak ada data atau metadata tentang asosiasi di dalam cluster di objek Kubernetes mana pun dan Anda tidak menambahkan anotasi apa pun ke akun layanan.

Prasyarat

- Sebuah kluster yang sudah ada. Jika Anda tidak memilikinya, Anda dapat membuatnya dengan mengikuti salah satu panduan di [Memulai](#).
- Prinsip IAM yang menciptakan asosiasi harus memiliki `iam:PassRole`.
- Versi terbaru dari AWS CLI diinstal dan dikonfigurasi pada perangkat Anda atau. AWS CloudShell Anda dapat memeriksa versi saat ini dengan `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Package manager seperti `yum` atau `apt-get`, atau Homebrew untuk macOS seringkali merupakan beberapa versi di belakang versi terbaru CLI. AWS Untuk menginstal versi terbaru, lihat [Menginstal](#) dan [Konfigurasi cepat dengan aws configure](#) di Panduan Pengguna Antarmuka Baris AWS Perintah. Versi AWS CLI yang diinstal di AWS CloudShell mungkin juga beberapa versi di

belakang versi terbaru. Untuk memperbaruinya, lihat [Menginstal AWS CLI ke direktori home Anda](#) di AWS CloudShell Panduan Pengguna.

- Alat baris `kubectl` perintah diinstal pada perangkat Anda atau AWS CloudShell. Versinya bisa sama dengan atau hingga satu versi minor lebih awal atau lebih lambat dari versi Kubernetes dari kluster Anda. Misalnya, jika versi cluster Anda `1.29`, Anda dapat menggunakan `kubectl` versi `1.28`, `1.29`, atau `1.30` dengan itu. Untuk menginstal atau memutakhirkan `kubectl`, lihat [the section called “Siapkan kubectl dan eksctl”](#).
- `kubectl` config file yang sudah ada yang berisi konfigurasi cluster Anda. Untuk membuat `kubectl` config file, lihat [the section called “Akses kluster dengan kubectl”](#).

Membuat asosiasi Pod Identity (AWS Console)

1. Buka [konsol Amazon EKS](#).
2. Di panel navigasi sebelah kiri, pilih Clusters, lalu pilih nama cluster yang ingin Anda konfigurasi untuk add-on EKS Pod Identity Agent.
3. Pilih tab Access.
4. Dalam asosiasi Pod Identity, pilih Create.
5. Untuk peran IAM, pilih peran IAM dengan izin yang Anda inginkan untuk memiliki beban kerja.

Note

Daftar ini hanya berisi peran yang memiliki kebijakan kepercayaan berikut yang memungkinkan EKS Pod Identity untuk menggunakannya.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
      "Effect": "Allow",
      "Principal": {
        "Service": "pods.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

```
    ]  
  }  
]  
}
```

`sts:AssumeRole`— EKS Pod Identity digunakan `AssumeRole` untuk mengambil peran IAM sebelum meneruskan kredensi sementara ke pod Anda.

`sts:TagSession`— EKS Pod Identity menggunakan `TagSession` untuk menyertakan tag sesi dalam permintaan ke AWS STS.

Anda dapat menggunakan tag ini di kunci kondisi dalam kebijakan kepercayaan untuk membatasi akun layanan, ruang nama, dan klaster mana yang dapat menggunakan peran ini.

Untuk daftar kunci kondisi Amazon EKS, lihat [Ketentuan yang ditentukan oleh Amazon Elastic Kubernetes Service di Referensi Otorisasi Layanan](#). Untuk mempelajari tindakan dan sumber daya yang dapat digunakan untuk menggunakan kunci kondisi, lihat [Tindakan yang ditentukan oleh Amazon Elastic Kubernetes Service](#).

6. Untuk namespace Kubernetes, pilih namespace Kubernetes yang berisi akun layanan dan beban kerja. Secara opsional, Anda dapat menentukan namespace dengan nama yang tidak ada di cluster.
7. Untuk akun layanan Kubernetes, pilih akun layanan Kubernetes yang akan digunakan. Manifes untuk beban kerja Kubernetes Anda harus menentukan akun layanan ini. Secara opsional, Anda dapat menentukan akun layanan berdasarkan nama yang tidak ada di cluster.
8. (Opsional) Untuk Tag, pilih Tambahkan tag untuk menambahkan metadata dalam pasangan kunci dan nilai. Tag ini diterapkan pada asosiasi dan dapat digunakan dalam kebijakan IAM.

Anda dapat mengulangi langkah ini untuk menambahkan beberapa tag.

9. Pilih Buat.

Membuat asosiasi Pod Identity (AWS CLI)

1. Jika Anda ingin mengaitkan kebijakan IAM yang ada dengan peran IAM Anda, lewati ke langkah berikutnya.

Buat kebijakan IAM. Anda dapat membuat kebijakan sendiri, atau menyalin kebijakan AWS terkelola yang telah memberikan beberapa izin yang Anda perlukan dan menyesuaikannya dengan

persyaratan spesifik Anda. Untuk informasi selengkapnya, lihat [Membuat kebijakan IAM](#) dalam Panduan Pengguna IAM.

- a. Buat file yang menyertakan izin untuk AWS layanan yang ingin diakses oleh Pod. Untuk daftar semua tindakan untuk semua AWS layanan, lihat [Referensi Otorisasi Layanan](#).

Anda dapat menjalankan perintah berikut untuk membuat contoh file kebijakan yang memungkinkan akses hanya-baca ke bucket Amazon S3. Anda dapat secara opsional menyimpan informasi konfigurasi atau skrip bootstrap di bucket ini, dan container di Pod Anda dapat membaca file dari bucket dan memuatnya ke dalam aplikasi Anda. Jika Anda ingin membuat kebijakan contoh ini, salin konten berikut ke perangkat Anda. Ganti *my-pod-secrets-bucket* dengan nama bucket Anda dan jalankan perintah.

```
cat >my-policy.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::my-pod-secrets-bucket"
    }
  ]
}
EOF
```

- b. Buat kebijakan IAM.

```
aws iam create-policy --policy-name my-policy --policy-document file:///my-policy.json
```

2. Buat peran IAM dan kaitkan dengan akun layanan Kubernetes.

- a. Jika Anda memiliki akun layanan Kubernetes yang ingin Anda ambil peran IAM, Anda dapat melewati langkah ini.

Buat akun layanan Kubernetes. Salin konten berikut ke perangkat Anda. Ganti *my-service-account* dengan nama yang Anda inginkan dan *default* dengan namespace yang berbeda, jika perlu. Jika Anda berubah *default*, namespace harus sudah ada.

```
cat >my-service-account.yaml <<EOF
apiVersion: v1
```

```
kind: ServiceAccount
metadata:
  name: my-service-account
  namespace: default
EOF
kubectl apply -f my-service-account.yaml
```

Jalankan perintah berikut.

```
kubectl apply -f my-service-account.yaml
```

- b. Jalankan perintah berikut untuk membuat file kebijakan kepercayaan untuk peran IAM.

```
cat >trust-relationship.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
      "Effect": "Allow",
      "Principal": {
        "Service": "pods.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
EOF
```

- c. Buat peran. Ganti *my-role* dengan nama untuk peran IAM Anda, dan *my-role-description* dengan deskripsi untuk peran Anda.

```
aws iam create-role --role-name my-role --assume-role-policy-document file://
trust-relationship.json --description "my-role-description"
```

- d. Lampirkan kebijakan IAM ke peran Anda. Ganti *my-role* dengan nama peran IAM Anda dan *my-policy* dengan nama kebijakan yang sudah ada yang Anda buat.

```
aws iam attach-role-policy --role-name my-role --policy-arn=arn:aws:iam::111122223333:policy/my-policy
```

 Note

Tidak seperti peran IAM untuk akun layanan, EKS Pod Identity tidak menggunakan anotasi pada akun layanan.

- e. Jalankan perintah berikut untuk membuat asosiasi. Ganti `my-cluster` dengan nama cluster, ganti `my-service-account` dengan nama yang Anda inginkan dan `default` dengan namespace yang berbeda, jika perlu.

```
aws eks create-pod-identity-association --cluster-name my-cluster --role-arn arn:aws:iam::111122223333:role/my-role --namespace default --service-account my-service-account
```

Contoh output adalah sebagai berikut.

```
{
  "association": {
    "clusterName": "my-cluster",
    "namespace": "default",
    "serviceAccount": "my-service-account",
    "roleArn": "arn:aws:iam::111122223333:role/my-role",
    "associationArn": "arn:aws:eks::111122223333:podidentityassociation/my-cluster/a-abcdefghijklmnop1",
    "associationId": "a-abcdefghijklmnop1",
    "tags": {},
    "createdAt": 1700862734.922,
    "modifiedAt": 1700862734.922
  }
}
```

 Note

Anda dapat menentukan namespace dan akun layanan berdasarkan nama yang tidak ada di cluster. Anda harus membuat namespace, akun layanan, dan beban kerja yang menggunakan akun layanan agar asosiasi Identitas Pod EKS berfungsi.

Konfirmasikan konfigurasi

1. Konfirmasikan bahwa kebijakan kepercayaan peran IAM telah dikonfigurasi dengan benar.

```
aws iam get-role --role-name my-role --query Role.AssumeRolePolicyDocument
```

Contoh output adalah sebagai berikut.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Allow EKS Auth service to assume this role for Pod Identities",
      "Effect": "Allow",
      "Principal": {
        "Service": "pods.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

2. Konfirmasikan bahwa kebijakan yang Anda lampirkan pada peran Anda di langkah sebelumnya dilampirkan pada peran tersebut.

```
aws iam list-attached-role-policies --role-name my-role --query
'AttachedPolicies[].PolicyArn' --output text
```

Contoh output adalah sebagai berikut.

```
arn:aws: iam::111122223333:policy/my-policy
```

3. Tetapkan variabel untuk menyimpan Nama Sumber Daya Amazon (ARN) kebijakan yang ingin Anda gunakan. Ganti *my-policy* dengan nama kebijakan yang ingin Anda konfirmasi izin.

```
export policy_arn=arn:aws: iam::111122223333:policy/my-policy
```

4. Lihat versi default kebijakan.

```
aws iam get-policy --policy-arn $policy_arn
```

Contoh output adalah sebagai berikut.

```
{
  "Policy": {
    "PolicyName": "my-policy",
    "PolicyId": "EXAMPLEBIOWGLDEXAMPLE",
    "Arn": "arn:aws:iam::111122223333:policy/my-policy",
    "Path": "/",
    "DefaultVersionId": "v1",
    [...]
  }
}
```

5. Lihat konten kebijakan untuk memastikan bahwa kebijakan tersebut menyertakan semua izin yang dibutuhkan Pod Anda. Jika perlu, ganti **1** dalam perintah berikut dengan versi yang dikembalikan pada output sebelumnya.

```
aws iam get-policy-version --policy-arn $policy_arn --version-id v1
```

Contoh output adalah sebagai berikut.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::my-pod-secrets-bucket"
    }
  ]
}
```

Jika Anda membuat kebijakan contoh di langkah sebelumnya, maka output Anda sama. Jika Anda membuat kebijakan yang berbeda, maka *example* kontennya berbeda.

Langkah Berikutnya

[the section called “Akun layanan Pod”](#)

Akses AWS Sumber Daya menggunakan Peran IAM Target Identitas EKS Pod

Saat menjalankan aplikasi di Amazon Elastic Kubernetes Service (Amazon EKS), Anda mungkin AWS perlu mengakses resource yang ada di akun yang sama atau berbeda. AWS Panduan ini menunjukkan kepada Anda cara mengatur akses antar akun ini menggunakan EKS Pod Identity, yang memungkinkan pod Kubernetes Anda mengakses sumber daya lainnya. AWS

Prasyarat

Sebelum Anda mulai, pastikan Anda telah menyelesaikan langkah-langkah berikut:

- [Siapkan Agen Identitas Amazon EKS Pod](#)
- [Buat peran Identitas Pod EKS](#)

Cara Kerjanya

Pod Identity memungkinkan aplikasi di kluster EKS Anda untuk mengakses AWS sumber daya di seluruh akun melalui proses yang disebut role chaining. Saat membuat asosiasi Pod Identity, Anda dapat menyediakan dua peran [IAM—peran Identitas Pod EKS](#) di akun yang sama dengan cluster EKS dan Peran IAM Target dari akun yang berisi AWS sumber daya Anda (seperti bucket S3 atau tabel DynamoDB). [Peran EKS Pod Identity](#) harus ada di akun kluster EKS Anda karena PassRole persyaratan [IAM](#), sedangkan Target IAM Role dapat berada di akun apa pun AWS . PassRole memungkinkan AWS entitas untuk mendelegasikan asumsi peran ke layanan lain. EKS Pod Identity digunakan PassRole untuk menghubungkan peran ke akun layanan Kubernetes, yang mengharuskan peran dan identitas yang melewatinya berada di AWS akun yang sama dengan cluster EKS. Ketika pod aplikasi Anda perlu mengakses AWS sumber daya, ia meminta kredensial dari Pod Identity. Pod Identity kemudian secara otomatis melakukan dua asumsi peran secara berurutan: pertama dengan asumsi peran [EKS Pod Identity, kemudian menggunakan kredensial-kredensialnya untuk mengasumsikan Target IAM Role](#). Proses ini memberi pod Anda kredensi sementara yang memiliki izin yang ditentukan dalam peran target, memungkinkan akses aman ke sumber daya di akun lain. AWS

Pertimbangan caching

Karena mekanisme caching, pembaruan peran IAM dalam asosiasi Identitas Pod yang ada mungkin tidak langsung berlaku di pod yang berjalan di kluster EKS Anda. Pod Identity Agent menyimpan

kredensial-kredensi IAM berdasarkan konfigurasi asosiasi pada saat kredensialnya diambil. Jika asosiasi hanya menyertakan ARN [peran Identitas Pod EKS](#) dan tidak ada Peran IAM Target, kredensial cache akan bertahan selama 6 jam. Jika asosiasi tersebut mencakup [peran EKS Pod Identity](#) ARN dan Target IAM Role, kredensial cache akan berlangsung selama 59 menit. Memodifikasi asosiasi yang ada, seperti memperbarui [peran EKS Pod Identity](#) ARN atau menambahkan Peran IAM Target, tidak mengatur ulang cache yang ada. Akibatnya, agen tidak akan mengenali pembaruan sampai kredensial yang di-cache disegarkan. Untuk menerapkan perubahan lebih cepat, Anda dapat membuat ulang pod yang ada; jika tidak, Anda harus menunggu cache kedaluwarsa.

Langkah 1: Buat dan kaitkan Peran IAM Target

Pada langkah ini, Anda akan membangun rantai kepercayaan yang aman dengan membuat dan mengonfigurasi Peran IAM Target. Untuk demonstrasi, kami akan membuat Peran IAM Target baru untuk membangun rantai kepercayaan antara dua AWS akun: peran [Identitas Pod EKS](#) (misalnya, `eks-pod-identity-primary-role`) di AWS akun kluster EKS mendapatkan izin untuk mengambil Peran IAM Target (misalnya `eks-pod-identity-aws-resources`) di akun target Anda, memungkinkan akses ke sumber daya seperti bucket AWS Amazon S3.

Buat Peran IAM Target

1. Buka [Amazon IAM console \(Konsol Amazon IAM\)](#).
2. Di bilah navigasi atas, verifikasi bahwa Anda masuk ke akun yang berisi AWS sumber daya (seperti bucket S3 atau tabel DynamoDB) untuk Peran IAM Target Anda.
3. Di panel navigasi sebelah kiri, pilih Peran.
4. Pilih tombol Buat peran, lalu AWS akun di bawah “Jenis entitas tepercaya.”
5. Pilih AWS Akun lain, masukkan nomor AWS akun Anda (akun tempat [peran Identitas EKS Pod](#) Anda ada), lalu pilih Berikutnya.
6. Tambahkan kebijakan izin yang ingin Anda kaitkan ke peran (misalnya, `AmazonS3FullAccess`), lalu pilih Berikutnya.
7. Masukkan nama peran, seperti `MyCustomIAMTargetRole`, lalu pilih Buat peran.

Perbarui kebijakan kepercayaan Peran IAM Target

1. Setelah membuat peran, Anda akan dikembalikan ke daftar Peran. Temukan dan pilih peran baru yang Anda buat di langkah sebelumnya (mis., `MyCustomIAMTargetRole`).
2. Pilih tab Trust relationship.
3. Klik Edit kebijakan kepercayaan di sisi kanan.

4. Di editor kebijakan, ganti JSON default dengan kebijakan kepercayaan Anda. Ganti nilai placeholder untuk nama peran dan 111122223333 ARN peran IAM dengan ID AWS akun yang menghosting kluster EKS Anda. Misalnya:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws: iam::111122223333:role/eks-pod-identity-primary-role"
      },
      "Action": "sts:AssumeRole"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws: iam::111122223333:role/eks-pod-identity-primary-role"
      },
      "Action": "sts:TagSession"
    }
  ]
}
```

Memperbarui kebijakan izin untuk peran EKS Pod Identity

Pada langkah ini, Anda akan memperbarui kebijakan izin [peran Identitas Pod EKS](#) yang terkait dengan kluster Amazon EKS Anda dengan menambahkan ARN Peran IAM Target sebagai sumber daya.

1. Buka [konsol Amazon EKS](#).
2. Di panel navigasi kiri, pilih Clusters, lalu pilih nama kluster EKS Anda.
3. Pilih tab Access.
4. Di bawah asosiasi Pod Identity, pilih [peran Identitas Pod EKS](#) Anda.
5. Pilih Izin, Tambahkan izin, lalu Buat kebijakan sebaris.
6. Pilih JSON di sisi kanan.
7. Di editor kebijakan, ganti JSON default dengan kebijakan izin Anda. Ganti nilai placeholder untuk nama peran dan 222233334444 ARN peran IAM dengan Peran IAM Target Anda. Misalnya:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ],
      "Resource": "arn:aws:iam::222233334444:role/eks-pod-identity-aws-
resources"
    }
  ]
}
```

Langkah 2: Kaitkan Peran IAM Target ke akun layanan Kubernetes

Pada langkah ini, Anda akan membuat asosiasi antara peran IAM Target dan akun layanan Kubernetes di kluster EKS Anda.

1. Buka [konsol Amazon EKS](#).
2. Di panel navigasi kiri, pilih Clusters, lalu pilih nama cluster yang ingin Anda tambahkan asosiasi.
3. Pilih tab Access.
4. Dalam asosiasi Pod Identity, pilih Create.
5. Pilih [peran Identitas Pod EKS dalam peran](#) IAM untuk diasumsikan oleh beban kerja Anda.
6. Pilih peran IAM Target dalam peran IAM Target yang akan diasumsikan oleh peran [EKS Pod Identity](#).
7. Di bidang namespace Kubernetes, masukkan nama namespace tempat Anda ingin membuat asosiasi (misalnya, my-app-namespace). Ini menentukan di mana akun layanan berada.
8. Di kolom akun layanan Kubernetes, masukkan nama akun layanan (misalnya, my-service-account) yang akan menggunakan kredensial IAM. Ini menautkan peran IAM ke akun layanan.
9. Pilih Buat untuk membuat asosiasi.

(Opsional) Langkah 3: Tambahkan Izin Eksternal ke Peran Target IAM

Terkadang, Anda mungkin perlu memberi pihak ketiga akses ke AWS sumber daya Anda (akses delegasi). Misalnya, Anda memutuskan untuk menyewa perusahaan pihak ketiga bernama Example

Corp untuk memantau AWS akun Anda dan membantu mengoptimalkan biaya. Untuk melacak pengeluaran harian Anda, Example Corp perlu mengakses AWS sumber daya Anda. Dalam hal ini, kami sarankan ExternalId untuk menambahkan kebijakan kepercayaan Peran Target IAM Anda untuk menghindari kemungkinan masalah [Deputi yang Bingung](#).

Edit kebijakan kepercayaan

1. Setelah membuat peran, Anda akan dikembalikan ke daftar Peran. Temukan dan klik peran baru yang Anda buat di langkah sebelumnya (mis., MyCustomIAMTargetRole).
2. Pilih tab Trust relationship.
3. Klik Edit kebijakan kepercayaan di sisi kanan.
4. Di editor kebijakan, ganti JSON default dengan kebijakan kepercayaan Anda. Ganti nilai ExternalId placeholder untuk `aws-region/other-account/cluster-name/namespace/service-account-name`, di mana "region" adalah wilayah klaster Anda, "111122223333" adalah ID AWS akun lainnya, "cluster-name" adalah nama cluster EKS, "namespace" adalah namespace Kubernetes, dan "" adalah nama akun layanan Kubernetes. AWS service-account-name Misalnya:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/eks-pod-identity-primary-role"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "sts:ExternalId": "region/111122223333/cluster-name/namespace/
service-account-name"
        }
      },
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/eks-pod-identity-primary-role"
      },
      "Action": "sts:TagSession"
    }
  ]
}
```

```
}
```

Konfigurasi Pod untuk mengakses AWS layanan dengan akun layanan

Jika sebuah Pod perlu mengakses AWS layanan, maka Anda harus mengonfigurasinya untuk menggunakan akun layanan Kubernetes. Akun layanan harus dikaitkan dengan peran AWS Identity and Access Management (IAM) yang memiliki izin untuk mengakses layanan AWS.

- Sebuah kluster yang sudah ada. Jika Anda tidak memilikinya, Anda dapat membuatnya menggunakan salah satu panduan di [Memulai](#).
- Akun layanan Kubernetes yang sudah ada dan asosiasi Identitas Pod EKS yang mengaitkan akun layanan dengan peran IAM. Peran tersebut harus memiliki kebijakan IAM terkait yang berisi izin yang Anda inginkan untuk Pod Anda untuk menggunakan AWS layanan. Untuk informasi selengkapnya tentang cara membuat akun dan peran layanan, dan mengonfigurasinya, lihat [the section called “Tetapkan peran IAM”](#).
- Versi terbaru dari AWS CLI diinstal dan dikonfigurasi pada perangkat Anda atau AWS CloudShell. Anda dapat memeriksa versi saat ini dengan `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Package manager seperti `yum` atau `apt-get`, atau Homebrew untuk macOS seringkali merupakan beberapa versi di belakang versi terbaru CLI AWS. Untuk menginstal versi terbaru, lihat [Menginstal](#) dan [Konfigurasi cepat dengan `aws configure`](#) di Panduan Pengguna Antarmuka Baris Perintah AWS. Versi AWS CLI yang diinstal di AWS CloudShell mungkin juga beberapa versi di belakang versi terbaru. Untuk memperbaruinya, lihat [Menginstal AWS CLI ke direktori home Anda](#) di AWS CloudShell Panduan Pengguna.
- Alat baris perintah `kubectl` diinstal pada perangkat Anda atau AWS CloudShell. Versinya bisa sama dengan atau hingga satu versi minor lebih awal atau lebih lambat dari versi Kubernetes dari kluster Anda. Misalnya, jika versi cluster Anda `1.29`, Anda dapat menggunakan `kubectl` versi `1.28`, `1.29`, atau `1.30` dengan itu. Untuk menginstal atau memutakhirkan `kubectl`, lihat [the section called “Siapkan `kubectl` dan `eksctl`”](#).
- `kubectl config` file yang sudah ada yang berisi konfigurasi cluster Anda. Untuk membuat `kubectl config` file, lihat [the section called “Akses kluster dengan `kubectl`”](#).
 1. Gunakan perintah berikut untuk membuat manifes deployment yang dapat digunakan untuk meng-deploy Pod untuk mengonfirmasi konfigurasi. Ganti *example values* dengan nilai-nilai milik Anda sendiri.

```
cat >my-deployment.yaml <<EOF
apiVersion: apps/v1
```

```
kind: Deployment
metadata:
  name: my-app
spec:
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      serviceAccountName: my-service-account
      containers:
      - name: my-app
        image: public.ecr.aws/nginx/nginx:X.XX
EOF
```

2. Terapkan manifes ke cluster Anda.

```
kubectl apply -f my-deployment.yaml
```

3. Konfirmasikan bahwa variabel lingkungan yang diperlukan ada untuk Pod Anda.

a. Lihat Pod yang di-deploy dengan deployment pada langkah sebelumnya.

```
kubectl get pods | grep my-app
```

Contoh output adalah sebagai berikut.

```
my-app-6f4dfff6cb-76cv9    1/1      Running    0          3m28s
```

b. Konfirmasikan bahwa Pod memiliki file token akun layanan mount.

```
kubectl describe pod my-app-6f4dfff6cb-76cv9 | grep
AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE:
```

Contoh output adalah sebagai berikut.

```
AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE: /var/run/secrets/
pods.eks.amazonaws.com/serviceaccount/eks-pod-identity-token
```

4. Konfirmasikan bahwa Pod Anda dapat berinteraksi dengan AWS layanan menggunakan izin yang Anda tetapkan dalam kebijakan IAM yang dilampirkan pada peran Anda.

 Note

Ketika sebuah Pod menggunakan AWS kredensial dari peran IAM yang terkait dengan akun layanan, AWS CLI atau lainnya SDKs dalam kontainer untuk Pod tersebut menggunakan kredensial yang disediakan oleh peran tersebut. Jika Anda tidak membatasi akses ke kredensial yang diberikan ke [peran IAM node Amazon EKS](#), Pod masih memiliki akses ke kredensial-kredensialnya. Untuk informasi selengkapnya, lihat [Membatasi akses ke profil instance yang ditetapkan ke node pekerja](#).

Jika Pod Anda tidak dapat berinteraksi dengan layanan seperti yang Anda harapkan, selesaikan langkah-langkah berikut untuk mengonfirmasi bahwa semuanya telah dikonfigurasi dengan benar.

- a. Konfirmasikan bahwa Pod Anda menggunakan versi AWS SDK yang mendukung asumsi peran IAM melalui asosiasi Identitas Pod EKS. Untuk informasi selengkapnya, lihat [the section called “Didukung SDKs”](#).
- b. Konfirmasikan bahwa penyebaran menggunakan akun layanan.

```
kubectl describe deployment my-app | grep "Service Account"
```

Contoh output adalah sebagai berikut.

```
Service Account: my-service-account
```

Berikan akses Pod ke AWS sumber daya berdasarkan tag

Kontrol akses berbasis atribut (ABAC) memberikan hak kepada pengguna melalui kebijakan yang menggabungkan atribut bersama-sama. EKS Pod Identity melampirkan tag ke kredensial sementara untuk setiap Pod dengan atribut seperti nama cluster, namespace, dan nama akun layanan. Tag sesi peran ini memungkinkan administrator untuk membuat peran tunggal yang dapat bekerja di seluruh akun layanan dengan mengizinkan akses ke AWS sumber daya berdasarkan tag yang cocok. Dengan menambahkan dukungan untuk tag sesi peran, Anda dapat menerapkan batasan keamanan

yang lebih ketat antara kluster, dan beban kerja dalam kluster, sambil menggunakan kembali peran IAM dan kebijakan IAM yang sama.

Contoh kebijakan dengan tag

Di bawah ini adalah contoh kebijakan IAM yang memberikan `s3:GetObject` izin ketika objek yang sesuai ditandai dengan nama cluster EKS.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectTagging"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "s3:ExistingObjectTag/eks-cluster-name": "${aws:PrincipalTag/eks-cluster-name}"
        }
      }
    }
  ]
}
```

Mengaktifkan atau menonaktifkan tag sesi

EKS Pod Identity menambahkan satu set tag sesi yang telah ditentukan sebelumnya ketika mengambil peran. Tag sesi ini memungkinkan administrator untuk membuat peran tunggal yang dapat bekerja di seluruh sumber daya dengan mengizinkan akses ke AWS sumber daya berdasarkan tag yang cocok.

Aktifkan tag sesi

Tag sesi diaktifkan secara otomatis dengan identitas Pod EKS — tidak ada tindakan yang diperlukan di pihak Anda. Secara default, EKS Pod Identity melampirkan satu set tag yang telah ditentukan ke sesi Anda. Untuk mereferensikan tag ini dalam kebijakan, gunakan sintaks yang `${aws:PrincipalTag/}` diikuti oleh kunci tag. Misalnya, `${aws:PrincipalTag/kubernetes-namespace}`.

- `eks-cluster-arn`
- `eks-cluster-name`
- `kubernetes-namespace`
- `kubernetes-service-account`
- `kubernetes-pod-name`
- `kubernetes-pod-uid`

Nonaktifkan tag sesi

AWS mengompres kebijakan sesi inline, kebijakan terkelola ARNs, dan tag sesi ke dalam format biner yang dikemas yang memiliki batas terpisah. Jika Anda menerima `PackedPolicyTooLarge` kesalahan yang menunjukkan format biner yang dikemas telah melebihi batas ukuran, Anda dapat mencoba mengurangi ukuran dengan menonaktifkan tag sesi yang ditambahkan oleh EKS Pod Identity. Untuk menonaktifkan tag sesi ini, ikuti langkah-langkah berikut:

1. Buka [konsol Amazon EKS](#).
2. Di panel navigasi kiri, pilih Cluster, lalu pilih nama cluster yang ingin Anda ubah.
3. Pilih tab Access.
4. Di asosiasi Identitas Pod, pilih ID asosiasi yang ingin Anda ubah di ID Asosiasi, lalu pilih Edit.
5. Di bawah Tag sesi, pilih Nonaktifkan tag sesi.
6. Pilih Simpan perubahan.

Tag lintas-akun

Semua tag sesi yang ditambahkan oleh EKS Pod Identity bersifat transitif; kunci dan nilai tag diteruskan ke `AssumeRole` tindakan apa pun yang digunakan beban kerja Anda untuk beralih peran ke akun lain. Anda dapat menggunakan tag ini dalam kebijakan di akun lain untuk membatasi akses

dalam skenario lintas akun. Untuk informasi selengkapnya, lihat [Merantai peran dengan tag sesi di Panduan Pengguna](#) IAM.

Tag kustom

EKS Pod Identity tidak dapat menambahkan tag kustom tambahan ke AssumeRole tindakan yang dilakukannya. Namun, tag yang Anda terapkan ke peran IAM selalu tersedia melalui format yang sama: `${aws:PrincipalTag/}` diikuti oleh kunci, misalnya `${aws:PrincipalTag/MyCustomTag}`.

Note

Tag yang ditambahkan ke sesi melalui `sts:AssumeRole` permintaan diutamakan dalam kasus konflik. Misalnya, katakan bahwa:

- Amazon EKS menambahkan kunci `eks-cluster-name` dan nilai `my-cluster` ke sesi ketika EKS mengasumsikan peran pelanggan dan
- Anda menambahkan `eks-cluster-name` tag ke peran IAM dengan nilai `my-own-cluster`.

Dalam hal ini, yang pertama diutamakan dan nilai untuk `eks-cluster-name` tag akan menjadi `my-cluster`

Menggunakan identitas pod dengan AWS SDK

Menggunakan kredensial Identitas Pod EKS

Untuk menggunakan kredensi dari asosiasi Identitas Pod EKS, kode Anda dapat menggunakan AWS SDK apa pun untuk membuat klien untuk AWS layanan dengan SDK, dan secara default SDK akan mencari di rantai lokasi untuk menggunakan kredensi AWS Identity and Access Management. Kredensial Identitas Pod EKS akan digunakan jika Anda tidak menentukan penyedia kredensi saat Anda membuat klien atau menginisialisasi SDK.

Ini berfungsi karena EKS Pod Identities telah ditambahkan ke penyedia kredensi Container yang dicari dalam satu langkah dalam rantai kredensi default. Jika beban kerja Anda saat ini menggunakan kredensial yang sebelumnya ada dalam rantai kredensial, kredensial-kredensialnya akan terus digunakan meskipun Anda mengonfigurasi asosiasi Identitas Pod EKS untuk beban kerja yang sama.

Untuk informasi selengkapnya tentang cara kerja Identitas Pod EKS, lihat [the section called “Cara kerjanya”](#).

Saat menggunakan [Pelajari cara EKS Pod Identity memberikan akses Pod ke AWS layanan](#), container dalam Pod Anda harus menggunakan versi AWS SDK yang mendukung asumsi peran IAM dari Agen Identitas Pod EKS. Pastikan Anda menggunakan versi berikut, atau yang lebih baru, untuk AWS SDK Anda:

- Java (Versi 2) - [2.21.30](#)
- Jawa — [1.12.746](#)
- [Pergi v1 - v1.47.11](#)
- Go v2 - [rilis-2023-11-14](#)
- [Python \(Boto3\) — 1.34.41](#)
- [Python \(botocore\) — 1.34.41](#)
- AWS [CLI — 1.30.0](#)

AWS [CLI — 2.15.0](#)

- JavaScript v2 — [2.1550.0](#)
- JavaScript [v3 — v3.458.0](#)
- [Kotlin — v1.0.1](#)
- [Ruby — 3.188.0](#)
- Karat — [rilis-2024-03-13](#)
- [C++ — 1.11.263](#)
- .NET — [3.7.734.0](#)
- PowerShell — [4.1.502](#)
- PHP — [3.287.1](#)

Untuk memastikan bahwa Anda menggunakan SDK yang didukung, ikuti petunjuk penginstalan SDK pilihan Anda di [Tools to Build on AWS](#) saat Anda membuat container.

Untuk daftar add-on yang mendukung EKS Pod Identity, lihat [the section called “Referensi Dukungan Pod Identity”](#).

Nonaktifkan IPv6 di Agen Identitas Pod EKS

AWS Management Console

1. Untuk menonaktifkan IPv6 di EKS Pod Identity Agent, tambahkan konfigurasi berikut ke pengaturan konfigurasi Opsional dari EKS Add-on.
 - a. Buka [konsol Amazon EKS](#).
 - b. Di panel navigasi kiri, pilih Clusters, lalu pilih nama cluster yang ingin Anda konfigurasi add-on.
 - c. Pilih tab Add-ons.
 - d. Pilih kotak di kanan atas kotak add-on Agen Identitas Pod EKS dan kemudian pilih Edit.
 - e. Pada halaman Configure EKS Pod Identity Agent:
 - i. Pilih Versi yang ingin Anda gunakan. Kami menyarankan Anda menyimpan versi yang sama dengan langkah sebelumnya, dan memperbarui versi dan konfigurasi dalam tindakan terpisah.
 - ii. Perluas pengaturan konfigurasi opsional.
 - iii. Masukkan kunci JSON **"agent"**: dan nilai objek JSON bersarang dengan kunci **"additionalArgs"**: dalam nilai Konfigurasi. Teks yang dihasilkan harus berupa objek JSON yang valid. Jika kunci dan nilai ini adalah satu-satunya data dalam kotak teks, kelilingi kunci dan nilai dengan kurawal kurawal. { } Contoh berikut menunjukkan kebijakan jaringan diaktifkan:

```
{
  "agent": {
    "additionalArgs": {
      "-b": "169.254.170.23"
    }
  }
}
```

Konfigurasi ini menetapkan IPv4 alamat menjadi satu-satunya alamat yang digunakan oleh agen.

- f. Untuk menerapkan konfigurasi baru dengan mengganti pod EKS Pod Identity Agent, pilih Simpan perubahan.

Amazon EKS menerapkan perubahan pada Add-on EKS dengan menggunakan peluncuran Kubernetes DaemonSet for EKS Pod Identity Agent. Anda dapat melacak status peluncuran

dalam riwayat Pembaruan add-on di dan dengan AWS Management Console . `kubectl rollout status daemonset/eks-pod-identity-agent --namespace kube-system`

`kubectl rollout` memiliki perintah berikut:

```
$ kubectl rollout

history  -- View rollout history
pause    -- Mark the provided resource as paused
restart  -- Restart a resource
resume   -- Resume a paused resource
status   -- Show the status of the rollout
undo     -- Undo a previous rollout
```

Jika peluncuran terlalu lama, Amazon EKS akan membatalkan peluncuran, dan pesan dengan jenis Pembaruan Addon dan status Gagal akan ditambahkan ke riwayat Pembaruan add-on. Untuk menyelidiki masalah apa pun, mulailah dari riwayat peluncuran, dan jalankan `kubectl logs` pada pod EKS Pod Identity Agent untuk melihat log dari EKS Pod Identity Agent.

2. Jika entri baru dalam riwayat Pembaruan memiliki status Sukses, maka peluncuran telah selesai dan add-on menggunakan konfigurasi baru di semua pod Agen Identitas Pod EKS.

AWS CLI

1. Untuk menonaktifkan IPv6 di EKS Pod Identity Agent, tambahkan konfigurasi berikut ke nilai konfigurasi Eks Add-on.

Jalankan perintah AWS CLI berikut. Ganti `my-cluster` dengan nama cluster Anda dan peran IAM ARN dengan peran yang Anda gunakan.

```
aws eks update-addon --cluster-name my-cluster --addon-name eks-pod-identity-agent \
  --resolve-conflicts PRESERVE --configuration-values '{"agent":{"additionalArgs":
{ "-b": "169.254.170.23"}}}'
```

Konfigurasi ini menetapkan IPv4 alamat menjadi satu-satunya alamat yang digunakan oleh agen.

Amazon EKS menerapkan perubahan pada Add-on EKS dengan menggunakan peluncuran Kubernetes DaemonSet for EKS Pod Identity Agent. Anda dapat melacak status peluncuran dalam riwayat Pembaruan add-on di dan dengan AWS Management Console . `kubectl rollout status daemonset/eks-pod-identity-agent --namespace kube-system`

kubectl rollout memiliki perintah berikut:

```
kubectl rollout

history -- View rollout history
pause   -- Mark the provided resource as paused
restart -- Restart a resource
resume  -- Resume a paused resource
status  -- Show the status of the rollout
undo    -- Undo a previous rollout
```

Jika peluncuran terlalu lama, Amazon EKS akan membatalkan peluncuran, dan pesan dengan jenis Pembaruan Addon dan status Gagal akan ditambahkan ke riwayat Pembaruan add-on. Untuk menyelidiki masalah apa pun, mulailah dari riwayat peluncuran, dan jalankan `kubectl logs` pada pod EKS Pod Identity Agent untuk melihat log dari EKS Pod Identity Agent.

Buat peran IAM dengan kebijakan kepercayaan yang diperlukan oleh EKS Pod Identity

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
      "Effect": "Allow",
      "Principal": {
        "Service": "pods.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

sts:AssumeRole

EKS Pod Identity menggunakan `AssumeRole` untuk mengambil peran IAM sebelum meneruskan kredensialnya sementara ke pod Anda.

sts:TagSession

EKS Pod Identity menggunakan TagSession untuk menyertakan tag sesi dalam permintaan ke AWS STS.

Anda dapat menggunakan tag ini di kunci kondisi dalam kebijakan kepercayaan untuk membatasi akun layanan, ruang nama, dan klaster mana yang dapat menggunakan peran ini.

Untuk daftar kunci kondisi Amazon EKS, lihat [Ketentuan yang ditentukan oleh Amazon Elastic Kubernetes Service di Referensi Otorisasi Layanan](#). Untuk mempelajari tindakan dan sumber daya yang dapat digunakan untuk menggunakan kunci kondisi, lihat [Tindakan yang ditentukan oleh Amazon Elastic Kubernetes Service](#).

Mengelola sumber daya komputasi dengan menggunakan node

Node Kubernetes adalah mesin yang menjalankan aplikasi kontainer. Setiap node memiliki komponen-komponen berikut:

- [Container runtime](#) — Perangkat lunak yang bertanggung jawab untuk menjalankan kontainer.
- [kubelet](#) — Memastikan bahwa kontainer sehat dan berjalan di dalam Pod yang terkait.
- [kube-proxy](#) — Mempertahankan aturan jaringan yang memungkinkan komunikasi ke Pod Anda.

Untuk informasi selengkapnya, lihat [Simpul](#) dalam dokumentasi Kubernetes.

Cluster Amazon EKS Anda dapat menjadwalkan Pod pada kombinasi node yang [dikelola EKS Auto Mode](#), node yang [dikelola sendiri](#), [grup node terkelola Amazon EKS](#), [AWS Fargate](#), dan [Amazon EKS Hybrid Nodes](#). Untuk mempelajari lebih lanjut tentang node yang digunakan di klaster Anda, lihat [the section called “Akses sumber daya klaster”](#).

Note

Tidak termasuk node hybrid, node harus berada di VPC yang sama dengan subnet yang Anda pilih saat Anda membuat cluster. Namun, node tidak harus berada di subnet yang sama.

Bandingkan opsi komputasi

Tabel berikut memberikan beberapa kriteria untuk dievaluasi saat memutuskan opsi mana yang paling sesuai dengan kebutuhan Anda. Node yang dikelola sendiri adalah opsi lain yang mendukung semua kriteria yang tercantum, tetapi mereka membutuhkan lebih banyak pemeliharaan manual. Untuk informasi selengkapnya, lihat [the section called “Simpul yang dikelola sendiri”](#).

Note

Bottlerocket memiliki beberapa perbedaan spesifik dari informasi umum dalam tabel ini. [Untuk informasi lebih lanjut, lihat dokumentasi Bottlerocket di GitHub](#)

Kriteria	Grup simpul terkelola EKS	Mode Otomatis EKS	Node Hibrida Amazon EKS
Dapat dikerahkan ke Outposts AWS	Tidak	Tidak	Tidak
Dapat dikerahkan ke Zona AWS Lokal	Ya	Tidak	Tidak
Dapat menjalankan kontainer yang memerlukan Windows	Ya	Tidak	Tidak
Dapat menjalankan kontainer yang membutuhkan Linux	Ya	Ya	Ya
Dapat menjalankan beban kerja yang memerlukan Chip Inferentia	Ya — Hanya untuk simpul Amazon Linux	Ya	Tidak
Dapat menjalankan beban kerja yang memerlukan GPU	Ya — Hanya untuk simpul Amazon Linux	Ya	Ya
Dapat menjalankan beban kerja yang memerlukan prosesor Arm	Ya	Ya	Ya
Dapat menjalankan AWS Bottlerocket	Ya	Ya	Tidak
Pod berbagi CPU, memori, penyimpanan, dan sumber daya	Ya	Ya	Ya

Kriteria	Grup simpul terkelola EKS	Mode Otomatis EKS	Node Hibrida Amazon EKS
jaringan dengan Pod lain.			
Harus menerapkan dan mengelola instans Amazon EC2	Ya	Tidak - Pelajari tentang instans EC2 terkelola	Ya — mesin fisik atau virtual lokal dikelola oleh Anda dengan alat pilihan Anda.
Harus mengamankan, memelihara, dan menambal sistem operasi EC2 instans Amazon	Ya	Tidak	Ya — sistem operasi yang berjalan pada mesin fisik atau virtual Anda dikelola oleh Anda dengan alat pilihan Anda.
Dapat memberikan argumen bootstrap pada penerapan node, seperti argumen kubelet ekstra.	Ya — Menggunakan eksctl atau meluncurkan template dengan AMI kustom.	Tidak - Gunakan a NodeClass untuk mengkonfigurasi node	Ya - Anda dapat menyesuaikan argumen bootstrap dengan nodeadm. Lihat the section called “Node hibrida nodeadm” .
Dapat menetapkan alamat IP ke Pod dari blok CIDR yang berbeda dari alamat IP yang ditetapkan ke node.	Ya - Menggunakan template peluncuran dengan AMI kustom. Untuk informasi selengkapnya, lihat the section called “Templat peluncuran” .	Tidak	Ya - lihat the section called “Konfigurasi CNI” .
Dapat SSH ke simpul	Ya	Tidak - Pelajari cara memecahkan masalah node	Ya

Kriteria	Grup simpul terkelola EKS	Mode Otomatis EKS	Node Hibrida Amazon EKS
Dapat men-deploy AMI kustom Anda sendiri ke simpul	Ya - Menggunakan template peluncuran	Tidak	Ya
Dapat men-deploy CNI kustom Anda sendiri ke simpul	Ya — Menggunakan templat peluncuran dengan AMI kustom	Tidak	Ya
Harus memperbarui node AMI sendiri	Ya - Jika Anda menerapkan AMI Amazon EKS yang dioptimalkan, Anda akan diberi tahu di konsol Amazon EKS saat pembaruan tersedia. Anda dapat melakukan pembaruan dengan satu klik di konsol. Jika Anda menerapkan AMI khusus, Anda tidak akan diberi tahu di konsol Amazon EKS saat pembaruan tersedia. Anda harus melakukan pembaruan sendiri.	Tidak	Ya - sistem operasi yang berjalan pada mesin fisik atau virtual Anda dikelola oleh Anda dengan perkakas pilihan Anda. Lihat the section called “Siapkan sistem operasi” .

Kriteria	Grup simpul terkelola EKS	Mode Otomatis EKS	Node Hibrida Amazon EKS
Harus memperbarui versi node Kubernetes sendiri	Ya - Jika Anda menerapkan AMI Amazon EKS yang dioptimalkan, Anda akan diberi tahu di konsol Amazon EKS saat pembaruan tersedia. Anda dapat melakukan pembaruan dengan satu klik di konsol. Jika Anda menerapkan AMI khusus, Anda tidak akan diberi tahu di konsol Amazon EKS saat pembaruan tersedia. Anda harus melakukan pembaruan sendiri.	Tidak	Ya - Anda mengelola peningkatan node hibrida dengan perkakas pilihan Anda sendiri atau dengan <code>nodeadm</code> . Lihat the section called “Tingkatkan node hibrida” .
Dapat menggunakan penyimpanan Amazon EBS dengan Pod	Ya	Ya, sebagai kemampuan terintegrasi. Pelajari cara membuat kelas penyimpanan .	Tidak
Dapat menggunakan penyimpanan Amazon EFS dengan Pod	Ya	Ya	Tidak
Dapat menggunakan Amazon FSx untuk penyimpanan Lustre dengan Pod	Ya	Ya	Tidak

Kriteria	Grup simpul terkelola EKS	Mode Otomatis EKS	Node Hibrida Amazon EKS
Dapat menggunakan Network Load Balancer untuk layanan	Ya	Ya	Ya - harus menggunakan tipe targetip.
Pods dapat berjalan dalam subnet publik	Ya	Ya	Tidak - pod berjalan di lingkungan lokal.
Dapat menetapkan grup keamanan VPC yang berbeda ke masing-masing Pod	Ya — Hanya untuk simpul Linux	Tidak	Tidak
Dapat menjalankan Kubernetes DaemonSets	Ya	Ya	Ya
Support HostPort dan HostNetwork dalam manifes Pod	Ya	Ya	Ya
AWS Ketersediaan wilayah	Semua wilayah yang didukung Amazon EKS	Semua wilayah yang didukung Amazon EKS	Semua wilayah yang didukung Amazon EKS kecuali Wilayah AWS GovCloud (AS) dan Wilayah Tiongkok.
Dapat menjalankan kontainer di host EC2 khusus Amazon	Ya	Tidak	Tidak

Kriteria	Grup simpul terkelola EKS	Mode Otomatis EKS	Node Hibrida Amazon EKS
Harga	Biaya EC2 instans Amazon yang menjalankan beberapa Pod. Untuk informasi selengkapnya, lihat EC2 harga Amazon .	Saat Mode Otomatis EKS diaktifkan di klaster, Anda membayar biaya terpisah, selain biaya EC2 instans standar, untuk instans yang diluncurkan menggunakan kemampuan komputasi Mode Otomatis. Jumlahnya bervariasi dengan jenis instans yang diluncurkan dan AWS wilayah tempat klaster Anda berada. Untuk informasi selengkapnya, lihat harga Amazon EKS .	Biaya node hybrid vCPU per jam. Untuk informasi selengkapnya, lihat harga Amazon EKS .

Sederhanakan siklus hidup node dengan grup node terkelola

Grup node terkelola Amazon EKS mengotomatiskan penyediaan dan pengelolaan siklus hidup node (instans Amazon EC2) untuk klaster Amazon EKS Kubernetes.

Dengan grup node terkelola Amazon EKS, Anda tidak perlu menyediakan atau mendaftarkan EC2 instans Amazon secara terpisah yang menyediakan kapasitas komputasi untuk menjalankan aplikasi Kubernetes Anda. Anda dapat membuat, memperbarui secara otomatis, atau mengakhiri simpul untuk klaster Anda dengan satu pengoperasian. Pembaruan dan penghentian node secara otomatis mengurus node untuk memastikan bahwa aplikasi Anda tetap tersedia.

Setiap node terkelola disediakan sebagai bagian dari grup Auto EC2 Scaling Amazon yang dikelola untuk Anda oleh Amazon EKS. Setiap sumber daya termasuk instans dan grup Auto Scaling berjalan

di dalam AWS akun Anda. Setiap grup node berjalan di beberapa Availability Zone yang Anda tentukan.

Grup node terkelola juga dapat secara opsional memanfaatkan perbaikan otomatis node, yang terus memantau kesehatan node. Secara otomatis bereaksi terhadap masalah yang terdeteksi dan menggantikan node bila memungkinkan. Ini membantu ketersediaan cluster secara keseluruhan dengan intervensi manual minimal. Untuk informasi selengkapnya, lihat [the section called “Kesehatan simpul”](#).

Anda dapat menambahkan grup node terkelola ke cluster baru atau yang sudah ada menggunakan konsol Amazon EKScct1, AWS CLI, API AWS , atau infrastruktur sebagai alat kode termasuk. AWS CloudFormation [Node yang diluncurkan sebagai bagian dari grup node terkelola secara otomatis ditandai untuk penemuan otomatis oleh Kubernetes Cluster Autoscaler](#). Anda dapat menggunakan grup simpul untuk menerapkan label Kubernetes ke simpul dan memperbaruinya kapan saja.

Tidak ada biaya tambahan untuk menggunakan grup node terkelola Amazon EKS, Anda hanya membayar AWS sumber daya yang Anda sediakan. Ini termasuk EC2 instans Amazon, volume Amazon EBS, jam kluster Amazon EKS, dan infrastruktur lainnya AWS . Tidak ada biaya minimum dan tidak ada komitmen di muka.

Untuk memulai dengan klaster Amazon EKS baru dan grup simpul terkelola, lihat [the section called “Buat cluster \(Konsol dan CLI\)”](#).

Untuk menambahkan grup simpul terkelola ke klaster yang ada, lihat [the section called “Buat”](#).

Konsep grup simpul terkelola

- Grup node terkelola Amazon EKS membuat dan mengelola EC2 instans Amazon untuk Anda.
- Setiap node terkelola disediakan sebagai bagian dari grup Auto EC2 Scaling Amazon yang dikelola untuk Anda oleh Amazon EKS. Selain itu, setiap sumber daya termasuk EC2 instans Amazon dan grup Auto Scaling berjalan di dalam AWS akun Anda.
- Grup Auto Scaling dari grup node terkelola mencakup setiap subnet yang Anda tentukan saat membuat grup.
- [Amazon EKS menandai resource grup node terkelola sehingga dikonfigurasi untuk menggunakan Kubernetes Cluster Autoscaler](#).

⚠ Important

Jika Anda menjalankan aplikasi stateful di beberapa Availability Zone yang didukung oleh volume Amazon EBS dan menggunakan Kubernetes [Cluster Autoscaler](#), Anda harus [mengonfigurasi beberapa grup node, masing-masing dicakup](#) ke satu Availability Zone. Selain itu, Anda harus mengaktifkan opsi fitur `--balance-similar-node-groups`.

- Anda dapat menggunakan templat peluncuran khusus untuk tingkat fleksibilitas dan penyesuaian yang lebih besar saat menerapkan node terkelola. Misalnya, Anda dapat menentukan kubelet argumen tambahan dan menggunakan AMI kustom. Untuk informasi selengkapnya, lihat [the section called “Templat peluncuran”](#). Jika Anda tidak menggunakan templat peluncuran kustom saat pertama kali membuat grup node terkelola, ada templat peluncuran yang dibuat secara otomatis. Jangan memodifikasi templat yang dibuat secara otomatis ini atau terjadi kesalahan secara manual.
- Amazon EKS mengikuti model tanggung jawab bersama untuk patch keamanan CVEs dan pada grup node terkelola. Ketika simpul terkelola menjalankan Amazon EKS yang dioptimalkan AMI, Amazon EKS bertanggung jawab untuk membangun versi patch AMI ketika bug atau masalah dilaporkan. Kita bisa memublikasikan perbaikan. Namun, Anda bertanggung jawab untuk menerapkan versi AMI yang ditambah ini ke grup node terkelola Anda. Saat node terkelola menjalankan AMI kustom, Anda bertanggung jawab untuk membuat versi AMI yang ditambah saat bug atau masalah dilaporkan dan kemudian menerapkan AMI. Untuk informasi selengkapnya, lihat [the section called “Perbarui”](#).
- Grup simpul terkelola Amazon EKS dapat diluncurkan di subnet publik maupun privat. Jika Anda meluncurkan grup simpul terkelola di subnet publik pada atau setelah 22 April 2020, subnet tersebut harus mengatur `MapPublicIpOnLaunch` ke BETUL agar instans berhasil bergabung dengan klaster. Jika subnet publik dibuat menggunakan `eksctl` atau [AWS CloudFormation templat penjual Amazon EKS](#) pada atau setelah 26 Maret 2020, maka pengaturan ini sudah disetel ke true. Jika subnet publik dibuat sebelum 26 Maret 2020, Anda harus mengubah pengaturan secara manual. Untuk informasi selengkapnya, lihat [Memodifikasi atribut IPv4 pengalamatan publik untuk subnet Anda](#).
- Saat menerapkan grup node terkelola di subnet pribadi, Anda harus memastikan bahwa grup tersebut dapat mengakses Amazon ECR untuk menarik gambar kontainer. [Anda dapat melakukan ini dengan menghubungkan gateway NAT ke tabel rute subnet atau dengan menambahkan titik akhir VPC berikut AWS PrivateLink :](#)
 - Antarmuka titik akhir API Amazon ECR - `com.amazonaws.region-code.ecr.api`

- Antarmuka titik akhir API registri Amazon ECR Docker - `com.amazonaws.region-code.ecr.dkr`
- Titik akhir gerbang Amazon S3 - `com.amazonaws.region-code.s3`

Untuk layanan dan titik akhir lainnya yang umum digunakan, lihat [the section called “klaster privat”](#)

- [Grup node terkelola tidak dapat digunakan di AWS Outposts atau di Wavelength.AWS](#) Grup node terkelola dapat dibuat di [AWS Local Zones](#). Untuk informasi selengkapnya, lihat [the section called “AWS Local Zones”](#).
- Anda dapat membuat beberapa grup simpul terkelola dalam satu klaster. Misalnya, Anda dapat membuat satu grup node dengan Amazon EKS standar yang dioptimalkan Amazon Linux AMI untuk beberapa beban kerja dan lainnya dengan varian GPU untuk beban kerja yang memerlukan dukungan GPU.
- Jika grup node terkelola mengalami kegagalan [pemeriksaan status EC2 instans Amazon](#), Amazon EKS mengembalikan kode kesalahan untuk membantu Anda mendiagnosis masalah. Untuk informasi selengkapnya, lihat [the section called “Kode kesalahan grup node terkelola”](#).
- Amazon EKS menambahkan label Kubernetes pada instans grup simpul terkelola. Amazon EKS yang diberi label ini diawali dengan `eks.amazonaws.com`.
- Amazon EKS otomatis membuang simpul menggunakan API Kubernetes selama pengakhiran atau pembaruan.
- Anggaran gangguan pod tidak dihormati saat mengakhiri sebuah node dengan AZRebalance atau mengurangi jumlah node yang diinginkan. Tindakan ini mencoba untuk mengusir Pod pada node. Tetapi jika dibutuhkan lebih dari 15 menit, node akan dihentikan terlepas dari apakah semua Pod pada node dihentikan. Untuk memperpanjang periode hingga node dihentikan, tambahkan hook siklus hidup ke grup Auto Scaling. Untuk informasi selengkapnya, lihat [Menambahkan kait siklus hidup](#) di Panduan Pengguna Amazon EC2 Auto Scaling.
- Untuk menjalankan proses drain dengan benar setelah menerima pemberitahuan gangguan Spot atau pemberitahuan penyeimbangan kapasitas, `CapacityRebalance` harus diatur ke `true`
- Memperbarui grup node terkelola akan menghormati anggaran gangguan Pod yang Anda tetapkan untuk Pod Anda. Untuk informasi selengkapnya, lihat [the section called “Perbarui detail perilaku”](#).
- Tidak ada biaya tambahan untuk menggunakan grup simpul terkelola Amazon EKS. Anda hanya membayar untuk AWS sumber daya yang Anda berikan.
- Jika Anda ingin mengenkripsi volume Amazon EBS untuk simpul Anda, Anda dapat men-deploy simpul menggunakan templat peluncuran. Untuk menerapkan node terkelola dengan volume Amazon EBS terenkripsi tanpa menggunakan templat peluncuran, enkripsi semua volume Amazon

EBS baru yang dibuat di akun Anda. Untuk informasi selengkapnya, lihat [Enkripsi secara default](#) di Panduan EC2 Pengguna Amazon.

Tipe kapasitas grup simpul terkelola

Saat membuat grup simpul terkelola, Anda dapat memilih tipe kapasitas Sesuai Permintaan atau Spot. Amazon EKS menerapkan grup node terkelola dengan grup EC2 Auto Scaling Amazon yang hanya berisi On-Demand atau hanya Instans Spot Amazon EC2. Anda dapat menjadwalkan Pod untuk aplikasi toleran kesalahan ke grup node yang dikelola Spot, dan kesalahan aplikasi yang tidak toleran pada grup node On-Demand dalam satu cluster Kubernetes. Secara default, grup node terkelola menerapkan instans Amazon EC2 On-Demand.

Sesuai Permintaan

Dengan Instans Sesuai Permintaan, Anda membayar kapasitas komputasi pada jam kedua tanpa komitmen jangka panjang.

Secara default, jika Anda tidak menentukan Tipe kapasitas, grup simpul terkelola disediakan dengan instans Sesuai Permintaan. Grup node terkelola mengonfigurasi grup EC2 Auto Scaling Amazon atas nama Anda dengan setelan berikut diterapkan:

- Strategi alokasi untuk penyediaan kapasitas Sesuai Permintaan diatur ke `prioritized`. Grup simpul terkelola menggunakan urutan tipe instans yang dilewatkan dalam API untuk menentukan tipe instans mana yang akan digunakan pertama kali ketika memenuhi kapasitas Sesuai Permintaan. Misalnya, Anda dapat menentukan tiga tipe instans dalam urutan berikut: `c5.large`, `c4.large`, dan `c3.large`. Ketika Instans Sesuai Permintaan Anda diluncurkan, grup simpul terkelola memenuhi kapasitas Sesuai Permintaan dengan memulai `c5.large`, lalu `c4.large`, dan kemudian `c3.large`. Untuk informasi selengkapnya, lihat [grup Amazon EC2 Auto Scaling](#) di Panduan Pengguna Amazon Auto EC2 Scaling.
- Amazon EKS menambahkan label Kubernetes berikut untuk semua simpul dalam grup simpul terkelola Anda yang menentukan tipe kapasitas: `eks.amazonaws.com/capacityType: ON_DEMAND`. Anda dapat menggunakan label ini untuk menjadwalkan aplikasi yang tidak menoleransi stateful atau kesalahan pada simpul Sesuai Permintaan.

Spot

Instans EC2 Spot Amazon adalah EC2 kapasitas Amazon cadangan yang menawarkan diskon tajam dari harga On-Demand. Instans EC2 Spot Amazon dapat diinterupsi dengan pemberitahuan interupsi

dua menit saat EC2 membutuhkan kapasitas kembali. Untuk informasi selengkapnya, lihat [Instans Spot](#) di Panduan EC2 Pengguna Amazon. Anda dapat mengonfigurasi grup node terkelola dengan Instans EC2 Spot Amazon untuk mengoptimalkan biaya node komputasi yang berjalan di kluster Amazon EKS Anda.

Untuk menggunakan Instans Spot di dalam grup node terkelola, buat grup node terkelola dengan menyetel tipe kapasitas sebagai `spot`. Grup node terkelola mengonfigurasi grup EC2 Auto Scaling Amazon atas nama Anda dengan praktik terbaik Spot berikut yang diterapkan:

- Untuk memastikan bahwa node Spot Anda disediakan dalam kumpulan kapasitas Spot yang optimal, strategi alokasi diatur ke salah satu dari berikut ini:
 - `price-capacity-optimized(PCO)` — Saat membuat grup node baru di cluster dengan versi Kubernetes 1.28 atau lebih tinggi, strategi alokasi diatur ke `price-capacity-optimized`. Namun, strategi alokasi tidak akan diubah untuk grup node yang sudah dibuat `capacity-optimized` sebelum grup node terkelola Amazon EKS mulai mendukung PCO.
 - `capacity-optimized(CO)` — Saat membuat grup node baru di cluster dengan versi Kubernetes 1.27 atau lebih rendah, strategi alokasi diatur ke `capacity-optimized`.

Untuk meningkatkan jumlah kumpulan kapasitas Spot yang tersedia untuk mengalokasikan kapasitas, konfigurasi grup node terkelola untuk menggunakan beberapa jenis instance.

- Amazon EC2 Spot Capacity Rebalancing diaktifkan sehingga Amazon EKS dapat mengurus dan menyeimbangkan kembali node Spot Anda dengan anggun untuk meminimalkan gangguan aplikasi saat node Spot berisiko tinggi mengalami gangguan. Untuk informasi selengkapnya, lihat [Penyeimbangan Kembali Kapasitas EC2 Auto Scaling Amazon di Panduan Pengguna](#) Amazon Auto Scaling. EC2
 - Saat node Spot menerima rekomendasi penyeimbangan ulang, Amazon EKS secara otomatis mencoba meluncurkan node Spot pengganti baru.
 - Jika pemberitahuan gangguan Spot dua menit tiba sebelum pengganti simpul Spot berada dalam status `Ready`, Amazon EKS mulai membuang simpul Spot yang menerima rekomendasi penyeimbangan kembali. Amazon EKS mengurus node dengan upaya terbaik. Akibatnya, tidak ada jaminan bahwa Amazon EKS akan menunggu node pengganti bergabung dengan cluster sebelum mengurus node yang ada.
 - Ketika pengganti simpul Spot node mengalami `bootstrapped` dan dalam status `Ready` pada Kubernetes, Amazon EKS menutup dan membuang simpul Spot yang menerima rekomendasi penyeimbangan kembali. Mengepung node Spot memastikan bahwa pengontrol layanan tidak mengirim permintaan baru ke node Spot ini. Hal ini juga menghapusnya dari daftar simpul Spot

yang sehat, aktif. Menguras node Spot memastikan bahwa Pod yang sedang berjalan diusir dengan anggun.

- Amazon EKS menambahkan label Kubernetes berikut pada semua simpul dalam grup simpul terkelola Anda yang menentukan tipe kapasitas: `eks.amazonaws.com/capacityType: SPOT`. Anda dapat menggunakan label ini untuk menjadwalkan aplikasi toleran kesalahan pada simpul Spot.

Ketika ingin memutuskan apakah akan men-deploy grup simpul dengan kapasitas Sesuai Permintaan atau Spot, Anda harus mempertimbangkan kondisi berikut:

- Instans Spot cocok untuk aplikasi tanpa kewarganegaraan, toleran kesalahan, dan fleksibel. Ini termasuk beban kerja pelatihan batch dan machine learning, data besar ETLs seperti Apache Spark, aplikasi pemrosesan antrian, dan titik akhir API stateless. Karena Spot adalah EC2 kapasitas Amazon cadangan, yang dapat berubah seiring waktu, kami menyarankan Anda menggunakan kapasitas Spot untuk beban kerja yang toleran interupsi. Lebih khusus lagi, kapasitas Spot cocok untuk beban kerja yang dapat mentolerir periode di mana kapasitas yang diperlukan tidak tersedia.
- Kami menyarankan Anda menggunakan On-Demand untuk aplikasi yang tidak toleran terhadap kesalahan. Ini termasuk alat manajemen cluster seperti alat pemantauan dan operasional, penerapan yang memerlukan `StatefulSets`, dan aplikasi stateful, seperti database.
- Untuk memaksimalkan ketersediaan aplikasi Anda saat menggunakan Instans Spot, sebaiknya konfigurasi grup simpul terkelola Spot untuk menggunakan beberapa tipe instans. Kami merekomendasikan agar Anda menerapkan aturan berikut ketika menggunakan beberapa tipe instans:
 - Dalam grup node terkelola, jika Anda menggunakan [Cluster Autoscaler](#), sebaiknya gunakan kumpulan tipe instans yang fleksibel dengan jumlah vCPU dan sumber daya memori yang sama. Ini untuk memastikan bahwa node dalam skala cluster Anda seperti yang diharapkan. Misalnya, jika Anda memerlukan empat v CPUs dan delapan memori GiB, gunakan `c3.xlarge`, `c4.xlarge`, `c5.xlarge`, `c5d.xlarge`, `c5a.xlarge`, `c5n.xlarge`, atau jenis instance serupa lainnya.
 - Untuk meningkatkan ketersediaan aplikasi, sebaiknya gunakan beberapa grup node terkelola Spot. Untuk ini, setiap grup harus menggunakan satu set tipe instance fleksibel yang memiliki vCPU dan sumber daya memori yang sama. Misalnya, jika Anda memerlukan memori 4 v CPUs dan 8 GiB, kami sarankan Anda membuat satu grup node terkelola dengan `c3.xlarge`, `c4.xlarge`, `c5.xlarge`, `c5d.xlarge`, `c5a.xlarge`, `c5n.xlarge`, atau

jenis instance serupa lainnya, dan grup node terkelola kedua dengan `m3.xlarge`, `m4.xlarge`, `m5.xlarge`, `m5d.xlarge`, `m5a.xlarge`, `m5n.xlarge` atau jenis instance serupa lainnya.

- Saat menerapkan grup node Anda dengan tipe kapasitas Spot yang menggunakan templat peluncuran khusus, gunakan API untuk meneruskan beberapa jenis instance. Jangan melewati satu jenis instance melalui template peluncuran. Untuk informasi selengkapnya tentang men-deploy grup simpul menggunakan templat peluncuran, lihat [the section called “Templat peluncuran”](#).

Buat grup node terkelola untuk kluster Anda

Topik ini menjelaskan bagaimana Anda dapat meluncurkan grup node terkelola Amazon EKS dari node yang mendaftar dengan kluster Amazon EKS Anda. Setelah simpul bergabung dengan kluster, Anda dapat men-deploy aplikasi Kubernetes pada mereka.

Jika ini adalah pertama kalinya Anda meluncurkan grup node terkelola Amazon EKS, kami sarankan Anda mengikuti salah satu panduan kami di [Memulai](#). Panduan ini menyediakan panduan untuk membuat cluster Amazon EKS dengan node.

Important

- Node Amazon EKS adalah EC2 instance Amazon standar. Anda ditagih berdasarkan EC2 harga Amazon normal. Untuk informasi selengkapnya, lihat [EC2 Harga Amazon](#).
- Anda tidak dapat membuat node terkelola di AWS Wilayah yang mengaktifkan AWS Outposts atau AWS Wavelength. Anda dapat membuat node yang dikelola sendiri sebagai gantinya. Lihat informasi selengkapnya di [the section called “Amazon Linux”](#), [the section called “Windows”](#), dan [the section called “Bottlerocket”](#). Anda juga dapat membuat grup node Amazon Linux yang dikelola sendiri di Outpost. Untuk informasi selengkapnya, lihat [the section called “Simpul”](#).
- Jika Anda tidak [menentukan ID AMI](#) untuk `bootstrap.sh` file yang disertakan dengan Linux atau Bottlerocket yang dioptimalkan Amazon EKS, grup node terkelola menerapkan angka maksimum pada nilai `maxPods` Untuk contoh dengan kurang dari 30 vCPUs, jumlah maksimumnya adalah 110. Untuk contoh dengan lebih besar dari 30 vCPUs, angka maksimum melompat ke 250. Angka-angka ini didasarkan pada [ambang skalabilitas Kubernetes dan pengaturan yang direkomendasikan oleh pengujian tim skalabilitas](#)

Amazon EKS internal. Untuk informasi selengkapnya, lihat [plugin Amazon VPC CNI meningkatkan pod per node membatasi](#) posting blog.

- Sebuah kluster Amazon EKS yang sudah ada. Untuk menyebarkan satu, lihat [the section called “Membuat kluster”](#).
- Peran IAM yang ada untuk digunakan node. Untuk membuatnya, lihat [the section called “IAM role simpel”](#). Jika peran ini tidak memiliki salah satu kebijakan untuk VPC CNI, peran terpisah yang mengikuti diperlukan untuk pod VPC CNI.
- (Opsional, tetapi disarankan) Plugin Amazon VPC CNI untuk add-on Kubernetes dikonfigurasi dengan peran IAM sendiri yang memiliki kebijakan IAM yang diperlukan yang melekat padanya. Untuk informasi selengkapnya, lihat [the section called “Konfigurasi untuk IRSA”](#).
- Keakraban dengan pertimbangan yang tercantum dalam [Pilih jenis instans EC2 node Amazon yang](#) optimal. Bergantung pada jenis instans yang Anda pilih, mungkin ada prasyarat tambahan untuk cluster dan VPC Anda.
- Untuk menambahkan grup node terkelola Windows, Anda harus terlebih dahulu mengaktifkan dukungan Windows untuk kluster Anda. Untuk informasi selengkapnya, lihat [the section called “Aktifkan dukungan Windows”](#).

Anda dapat membuat grup node terkelola dengan salah satu dari berikut ini:

- [the section called “eksctl”](#)
- [the section called “AWS Management Console”](#)

eksctl

Buat grup node terkelola dengan eksctl

Prosedur ini membutuhkan eksctl versi 0.210.0 atau yang lebih baru. Anda dapat memeriksa versi Anda dengan perintah berikut:

```
eksctl version
```

Untuk petunjuk tentang cara menginstal atau meningkatkan eksctl, lihat [Instalasi](#) dalam eksctl dokumentasi.

1. (Opsional) Jika kebijakan IAM terkelola `AmazonEKS_CNI_Policy` dilampirkan ke peran IAM [node Amazon EKS Anda](#), kami sarankan untuk menetakannya ke peran IAM yang Anda kaitkan ke akun layanan Kubernetes sebagai gantinya. `aws-node` Untuk informasi selengkapnya, lihat [the section called “Konfigurasi untuk IRSA”](#).
2. Buat grup node terkelola dengan atau tanpa menggunakan templat peluncuran kustom. Menentukan template peluncuran secara manual memungkinkan kustomisasi grup node yang lebih besar. Misalnya, ini dapat memungkinkan penerapan AMI khusus atau memberikan argumen ke `bootstrap.sh` skrip di AMI Amazon EKS yang dioptimalkan. Untuk daftar lengkap setiap opsi yang tersedia dan default, masukkan perintah berikut.

```
eksctl create nodegroup --help
```

Dalam perintah berikut, ganti *my-cluster* dengan nama cluster Anda dan ganti *my-mng* dengan nama grup node Anda. Nama grup node tidak boleh lebih dari 63 karakter. Itu harus dimulai dengan huruf atau digit, tetapi juga dapat menyertakan tanda hubung dan garis bawah untuk karakter yang tersisa.

Important

Jika Anda tidak menggunakan templat peluncuran kustom saat pertama kali membuat grup node terkelola, jangan gunakan satu di lain waktu untuk grup node. Jika Anda tidak menentukan templat peluncuran kustom, sistem akan secara otomatis membuat templat peluncuran yang tidak kami sarankan agar Anda memodifikasi secara manual. Memodifikasi templat peluncuran yang dibuat secara otomatis ini secara manual dapat menyebabkan kesalahan.

Tanpa template peluncuran

`eksctl` membuat template EC2 peluncuran Amazon default di akun Anda dan menyebarkan grup node menggunakan template peluncuran yang dibuat berdasarkan opsi yang Anda tentukan. Sebelum menentukan nilai untuk `--node-type`, lihat [the section called “Jenis EC2 instans Amazon”](#).

Ganti *ami-family* dengan kata kunci yang diizinkan. Untuk informasi selengkapnya, lihat [Menyetel simpul AMI Family](#) dalam `eksctl` dokumentasi. Ganti *my-key* dengan nama Amazon EC2 key pair atau public key Anda. Kunci ini digunakan untuk SSH ke simpul Anda setelah diluncurkan.

Note

Untuk Windows, perintah ini tidak mengaktifkan SSH. Sebagai gantinya, ini mengaitkan EC2 key pair Amazon Anda dengan instance dan memungkinkan Anda untuk RDP ke dalam instance.

Jika Anda belum memiliki EC2 key pair Amazon, Anda dapat membuatnya di AWS Management Console. Untuk informasi Linux, lihat [pasangan EC2 kunci Amazon dan instans Linux](#) di Panduan EC2 Pengguna Amazon. Untuk informasi Windows, lihat [pasangan EC2 kunci Amazon dan instans Windows](#) di Panduan EC2 Pengguna Amazon.

Sebaiknya blokir akses Pod ke IMDS jika kondisi berikut benar:

- Anda berencana untuk menetapkan peran IAM ke semua akun layanan Kubernetes Anda sehingga Pod hanya memiliki izin minimum yang mereka butuhkan.
- Tidak ada Pod dalam kluster yang memerlukan akses ke layanan metadata EC2 instans Amazon (IMDS) karena alasan lain, seperti mengambil Region saat ini. AWS

Untuk informasi selengkapnya, lihat [Membatasi akses ke profil instance yang ditetapkan ke node pekerja](#).

Jika Anda ingin memblokir akses Pod ke IMDS, tambahkan `--disable-pod-imds` opsi ke perintah berikut.

```
eksctl create nodegroup \  
  --cluster my-cluster \  
  --region region-code \  
  --name my-mng \  
  --node-ami-family ami-family \  
  --node-type m5.large \  
  --nodes 3 \  
  --nodes-min 2 \  
  --nodes-max 4 \  
  --ssh-access \  
  --ssh-public-key my-key
```

Instance Anda secara opsional dapat menetapkan jumlah alamat IP yang jauh lebih tinggi ke Pod, menetapkan alamat IP ke Pod dari blok CIDR yang berbeda dari instans, dan di-deploy ke kluster

tanpa akses internet. Untuk informasi selengkapnya [the section called “Meningkatkan alamat IP”](#), lihat [the section called “Jaringan kustom”](#), dan [the section called “klaster privat”](#) untuk opsi tambahan untuk ditambahkan ke perintah sebelumnya.

Grup node terkelola menghitung dan menerapkan satu nilai untuk jumlah maksimum Pod yang dapat dijalankan pada setiap node grup node Anda, berdasarkan tipe instance. Jika Anda membuat grup node dengan tipe instans yang berbeda, nilai terkecil yang dihitung di semua tipe instance diterapkan sebagai jumlah maksimum Pod yang dapat dijalankan pada setiap tipe instance dalam grup node. Grup node terkelola menghitung nilai menggunakan skrip yang direferensikan di [Pod maksimum yang direkomendasikan Amazon EKS untuk setiap jenis EC2 instans Amazon](#).

Dengan template peluncuran

Template peluncuran harus sudah ada dan harus memenuhi persyaratan yang ditentukan dalam [Dasar-dasar konfigurasi template Luncurkan](#). Sebaiknya blokir akses Pod ke IMDS jika kondisi berikut benar:

- Anda berencana untuk menetapkan peran IAM ke semua akun layanan Kubernetes Anda sehingga Pod hanya memiliki izin minimum yang mereka butuhkan.
- Tidak ada Pod dalam klaster yang memerlukan akses ke layanan metadata EC2 instans Amazon (IMDS) karena alasan lain, seperti mengambil Region saat ini. AWS

Untuk informasi selengkapnya, lihat [Membatasi akses ke profil instance yang ditetapkan ke node pekerja](#).

Jika Anda ingin memblokir akses Pod ke IMDS, maka tentukan pengaturan yang diperlukan dalam template peluncuran.

- a. Salin konten berikut ke perangkat Anda. Ganti *example values* dan kemudian jalankan perintah yang dimodifikasi untuk membuat `eks-nodegroup.yaml` file. Beberapa pengaturan yang Anda tentukan saat men-deploy tanpa templat peluncuran dipindahkan ke templat peluncuran. Jika Anda tidak menentukan `version`, versi default template akan digunakan.

```
cat >eks-nodegroup.yaml <<EOF
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: my-cluster
  region: region-code
managedNodeGroups:
```

```
- name: my-mng
  launchTemplate:
    id: lt-id
    version: "1"
EOF
```

Untuk daftar selengkapnya tentang eksctl pengaturan file konfigurasi, lihat [Skema file Config](#) di dokumentasi eksctl. Instance Anda secara opsional dapat menetapkan jumlah alamat IP yang jauh lebih tinggi ke Pod, menetapkan alamat IP ke Pod dari blok CIDR yang berbeda dari instans, dan di-deploy ke kluster tanpa akses internet keluar. Untuk informasi selengkapnya [the section called “Meningkatkan alamat IP”](#), lihat [the section called “Jaringan kustom”](#), dan [the section called “kluster privat”](#) untuk opsi tambahan untuk ditambahkan ke file konfigurasi.

Jika Anda tidak menentukan ID AMI dalam template peluncuran, grup node terkelola menghitung dan menerapkan satu nilai untuk jumlah maksimum Pod yang dapat dijalankan pada setiap node grup node Anda, berdasarkan jenis instance. Jika Anda membuat grup node dengan tipe instans yang berbeda, nilai terkecil yang dihitung di semua tipe instance diterapkan sebagai jumlah maksimum Pod yang dapat dijalankan pada setiap tipe instance dalam grup node. Grup node terkelola menghitung nilai menggunakan skrip yang direferensikan di [Pod maksimum yang direkomendasikan Amazon EKS untuk setiap jenis EC2 instans Amazon](#).

Jika Anda menetapkan ID AMI di template peluncuran, tentukan jumlah maksimum Pod yang dapat berjalan di setiap node grup node jika Anda menggunakan [jaringan khusus](#) atau ingin [menambah jumlah alamat IP yang ditetapkan ke instans Anda](#). Untuk informasi selengkapnya, lihat [the section called “Amazon EKS merekomendasikan Pod maksimum untuk setiap jenis EC2 instans Amazon”](#).

b. Men-deploy grup simpul dengan perintah berikut.

```
eksctl create nodegroup --config-file eks-nodegroup.yaml
```

AWS Management Console

Buat grup node terkelola menggunakan AWS Management Console

1. Tunggu status kluster Anda ditampilkan sebagai ACTIVE. Anda tidak dapat membuat grup node terkelola untuk kluster yang belum ada ACTIVE.
2. Buka [konsol Amazon EKS](#).

3. Pilih nama cluster tempat Anda ingin membuat grup node terkelola.
4. Pilih tab Compute.
5. Pilih Tambahkan grup simpul.
6. Pada halaman Konfigurasi grup simpul, isi parameter yang sesuai, dan kemudian pilih Selanjutnya.
 - Nama — Masukkan nama unik untuk grup simpul terkelola Anda. Nama grup node tidak boleh lebih dari 63 karakter. Itu harus dimulai dengan huruf atau digit, tetapi juga dapat menyertakan tanda hubung dan garis bawah untuk karakter yang tersisa.
 - Peran IAM Node — Pilih peran instance node yang akan digunakan dengan grup node Anda. Untuk informasi selengkapnya, lihat [the section called “IAM role simpul”](#).

 Important

- Anda tidak dapat menggunakan peran yang sama yang digunakan untuk membuat cluster apa pun.
 - Sebaiknya gunakan peran yang saat ini tidak digunakan oleh grup node yang dikelola sendiri. Jika tidak, Anda berencana untuk menggunakan dengan grup node yang dikelola sendiri yang baru. Untuk informasi selengkapnya, lihat [the section called “Hapus”](#).
-
- Gunakan template peluncuran - (Opsional) Pilih jika Anda ingin menggunakan template peluncuran yang ada. Pilih Nama Template Luncurkan. Kemudian, pilih versi Template Luncurkan. Jika Anda tidak memilih versi, Amazon EKS menggunakan versi default template. Template peluncuran memungkinkan lebih banyak penyesuaian grup node Anda, seperti memungkinkan Anda untuk menerapkan AMI kustom, menetapkan jumlah alamat IP yang jauh lebih tinggi ke Pod, menetapkan alamat IP ke Pod dari blok CIDR yang berbeda dari instans, dan menerapkan node ke cluster tanpa akses internet keluar. Lihat informasi selengkapnya di [the section called “Meningkatkan alamat IP”](#), [the section called “Jaringan kustom”](#), dan [the section called “klaster privat”](#).

Template peluncuran harus memenuhi persyaratan di [Sesuaikan node terkelola dengan templat peluncuran](#). Jika Anda tidak menggunakan template peluncuran Anda sendiri, Amazon EKS API akan membuat template EC2 peluncuran Amazon default di akun Anda dan menyebarkan grup node menggunakan templat peluncuran default.

Jika Anda menerapkan [peran IAM untuk akun layanan](#), menetapkan izin yang diperlukan secara langsung ke setiap Pod yang memerlukan akses ke AWS layanan, dan tidak ada Pod di kluster Anda yang memerlukan akses ke IMDS karena alasan lain, seperti mengambil AWS Region saat ini, maka Anda juga dapat menonaktifkan akses ke IMDS untuk Pod yang tidak menggunakan jaringan host dalam template peluncuran. Untuk informasi selengkapnya, lihat [Membatasi akses ke profil instance yang ditetapkan ke node pekerja](#).

- Label Kubernetes — (Opsional) Anda dapat memilih untuk menerapkan label Kubernetes ke simpul dalam grup simpul terkelola Anda.
 - Kubernetes taints — (Opsional) Anda dapat memilih untuk menerapkan taints Kubernetes ke node di grup node terkelola Anda. Pilihan yang tersedia di menu Effect adalah **NoSchedule**, **NoExecute**, dan **PreferNoSchedule**. Untuk informasi selengkapnya, lihat [the section called “Node GPU node”](#).
 - Tanda – (Opsional) Anda dapat memilih untuk menandai grup simpul terkelola Amazon EKS Anda. Tag ini tidak menyebar ke sumber daya lain di grup node, seperti grup Auto Scaling atau instance. Untuk informasi selengkapnya, lihat [the section called “Pemberian tag pada sumber daya Anda”](#).
7. Pada halaman Atur konfigurasi komputasi dan penskalaan, isi parameter yang sesuai, dan kemudian pilih Selanjutnya.
- Jenis AMI - Pilih tipe AMI. Jika Anda menerapkan instance Arm, pastikan untuk meninjau pertimbangan di Amazon [EKS yang dioptimalkan Arm Amazon Linux AMIs](#) sebelum menerapkan.

Jika Anda menentukan template peluncuran di halaman sebelumnya, dan menetapkan AMI di template peluncuran, maka Anda tidak dapat memilih nilai. Nilai dari templat ditampilkan. AMI yang ditentukan dalam template harus memenuhi persyaratan dalam [Menentukan AMI](#).

- Tipe kapasitas – Pilih tipe kapasitas. Untuk informasi selengkapnya tentang memilih tipe kapasitas, lihat [the section called “Tipe kapasitas grup simpul terkelola”](#). Anda tidak dapat mencampur tipe kapasitas yang berbeda dalam grup node yang sama. Jika Anda ingin menggunakan kedua tipe kapasitas, buatlah grup simpul terpisah, masing-masing dengan tipe kapasitas dan instans mereka sendiri. Lihat [the section called “Reservasi GPUs untuk MNG”](#) untuk informasi tentang penyediaan dan penskalaan node pekerja yang dipercepat GPU.
- Jenis instans - Secara default, satu atau lebih jenis instance ditentukan. Untuk menghapus tipe instans default, pilih X di sisi kanan tipe instans. Pilih tipe instans yang akan digunakan dalam grup simpul terkelola. Untuk informasi selengkapnya, lihat [the section called “Jenis EC2 instans Amazon”](#).

Konsol menampilkan satu set tipe instans yang umum digunakan. Jika Anda perlu membuat grup node terkelola dengan tipe instance yang tidak ditampilkan, gunakan `eksctl`, AWS CLI, AWS CloudFormation, atau SDK untuk membuat grup node. Jika Anda menentukan template peluncuran pada halaman sebelumnya, maka Anda tidak dapat memilih nilai karena jenis instance harus ditentukan dalam template peluncuran. Nilai dari templat peluncuran ditampilkan. Jika Anda memilih Spot untuk tipe Kapasitas, maka kami merekomendasikan untuk menentukan beberapa tipe instans untuk meningkatkan ketersediaan.

- Ukuran disk — Masukkan ukuran disk (dalam GiB) yang akan digunakan untuk volume root node Anda.

Jika Anda menentukan template peluncuran di halaman sebelumnya, maka Anda tidak dapat memilih nilai karena harus ditentukan dalam template peluncuran.

- Ukuran yang diinginkan – Tentukan jumlah simpul yang harus dipertahankan oleh grup simpul terkelola saat peluncuran.

 Note

Amazon EKS tidak secara otomatis menskalakan grup node Anda masuk atau keluar. Namun, Anda dapat mengkonfigurasi Kubernetes Cluster Autoscaler untuk melakukan ini untuk Anda. Untuk informasi selengkapnya, lihat [Cluster Autoscaler](#) di AWS.

- Ukuran minimum – Tentukan jumlah simpul minimum yang dapat diskalakan ke dalam oleh grup simpul terkelola.
- Ukuran maksimum – Tentukan jumlah simpul maksimum yang dapat diskalakan keluar oleh grup simpul terkelola.
- Konfigurasi pembaruan grup node — (Opsional) Anda dapat memilih jumlah atau persentase node yang akan diperbarui secara paralel. Node ini tidak akan tersedia selama pembaruan. Untuk Maksimum tidak tersedia, pilih salah satu opsi berikut dan tentukan Nilai:
 - Angka — Pilih dan tentukan jumlah node dalam grup node Anda yang dapat diperbarui secara paralel.
 - Persentase — Pilih dan tentukan persentase node dalam grup node Anda yang dapat diperbarui secara paralel. Ini berguna jika Anda memiliki sejumlah besar node di grup node Anda.
- Konfigurasi perbaikan otomatis node — (Opsional) Jika Anda mengaktifkan kotak centang Aktifkan perbaikan otomatis node, Amazon EKS akan secara otomatis mengganti node

saat masalah yang terdeteksi terjadi. Untuk informasi selengkapnya, lihat [the section called “Kesehatan simpul”](#).

8. Di halaman Tentukan jaringan, isi parameter yang sesuai, dan pilih Selanjutnya.

- Subnet — Pilih subnet untuk meluncurkan simpul terkelola Anda.

 Important

Jika Anda menjalankan aplikasi stateful di beberapa Availability Zone yang didukung oleh volume Amazon EBS dan menggunakan Kubernetes [Cluster Autoscaler](#), Anda [harus mengonfigurasi beberapa grup node, masing-masing dicakup](#) ke satu Availability Zone. Selain itu, Anda harus mengaktifkan fitur `--balance-similar-node-groups`.

 Important

- Jika Anda memilih subnet publik, dan klaster Anda hanya memiliki titik akhir server API publik yang diaktifkan, maka subnet harus mengatur `MapPublicIPOnLaunch` ke `true` agar instans dapat bergabung dengan klaster. Jika subnet dibuat menggunakan `eksctl` atau [AWS CloudFormation templat penjual Amazon EKS](#) pada atau setelah 26 Maret 2020, maka pengaturan ini sudah diatur ke `true`. Jika subnet dibuat dengan `eksctl` atau AWS CloudFormation templat sebelum 26 Maret 2020, maka Anda perlu mengubah pengaturan secara manual. Untuk informasi selengkapnya, lihat [Memodifikasi atribut IPv4 pengalamatan publik untuk subnet Anda](#).
 - Jika Anda menggunakan templat peluncuran dan menentukan beberapa antarmuka jaringan, Amazon tidak EC2 akan menetapkan IPv4 alamat publik secara otomatis, meskipun `MapPublicIpOnLaunch` disetel ke `true`. Agar node dapat bergabung dengan cluster dalam skenario ini, Anda harus mengaktifkan titik akhir server API pribadi cluster, atau meluncurkan node di subnet pribadi dengan akses internet keluar yang disediakan melalui metode alternatif, seperti NAT Gateway. Untuk informasi selengkapnya, lihat [Pengalamatan IP EC2 instans Amazon](#) di Panduan EC2 Pengguna Amazon.
- Konfigurasikan akses SSH ke node (Opsional). Mengaktifkan SSH memungkinkan Anda untuk tekoneksi ke instans dan mengumpulkan informasi diagnostik jika ada masalah. Kami sangat menyarankan untuk mengaktifkan akses jarak jauh saat Anda membuat grup node. Anda tidak dapat mengaktifkan akses jarak jauh setelah grup node dibuat.

Jika Anda memilih untuk menggunakan template peluncuran, maka opsi ini tidak ditampilkan. Untuk mengaktifkan akses jarak jauh ke simpul Anda, tentukan pasangan kunci di templat peluncuran dan pastikan bahwa port yang tepat terbuka untuk simpul dalam grup keamanan yang Anda tentukan di templat peluncuran. Untuk informasi selengkapnya, lihat [the section called “Menggunakan grup keamanan kustom”](#).

 Note

Untuk Windows, perintah ini tidak mengaktifkan SSH. Sebagai gantinya, ini mengaitkan EC2 key pair Amazon Anda dengan instance dan memungkinkan Anda untuk RDP ke dalam instance.

- Untuk key pair SSH (Opsional), pilih kunci EC2 SSH Amazon yang akan digunakan. Untuk informasi Linux, lihat [pasangan EC2 kunci Amazon dan instans Linux](#) di Panduan EC2 Pengguna Amazon. Untuk informasi Windows, lihat [pasangan EC2 kunci Amazon dan instans Windows](#) di Panduan EC2 Pengguna Amazon. Jika Anda memilih untuk menggunakan template peluncuran, maka Anda tidak dapat memilih salah satu. Ketika kunci EC2 SSH Amazon disediakan untuk grup node menggunakan Bottlerocket AMIs, wadah administratif juga diaktifkan. Untuk informasi selengkapnya, lihat [Kontainer admin](#) di GitHub.
 - Untuk Izinkan akses jarak jauh SSH dari, jika Anda ingin membatasi akses ke instance tertentu, pilih grup keamanan yang terkait dengan instance tersebut. Jika Anda tidak memilih grup keamanan tertentu, maka akses SSH diizinkan dari mana saja di internet (0.0.0.0/0).
9. Pada halaman Tinjauan dan buat, tinjau konfigurasi grup simpul terkelola dan pilih Buat.

Jika node gagal bergabung dengan cluster, maka lihat [the section called “Simpul gagal untuk bergabung dengan klaster”](#) di chapter Troubleshooting.

10. Perhatikan status simpul Anda dan tunggu sampai simpul mencapai Status Ready.

```
kubectl get nodes --watch
```

11. (Hanya node GPU) Jika Anda memilih jenis instans GPU dan AMI akselerasi Amazon EKS yang dioptimalkan, maka Anda harus menerapkan [plugin perangkat NVIDIA untuk Kubernetes](#) sebagai a di cluster Anda. DaemonSet Ganti `vX.X.X` dengan s-device-plugin versi [NVIDIA/K8](#) yang Anda inginkan sebelum menjalankan perintah berikut.


```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/vX.X.X/
deployments/static/nvidia-device-plugin.yml
```

Instal add-on Kubernetes

Sekarang setelah Anda memiliki kluster Amazon EKS yang berfungsi dengan node, Anda siap untuk mulai menginstal add-on Kubernetes dan menerapkan aplikasi ke kluster Anda. Topik dokumentasi berikut membantu Anda untuk memperluas fungsionalitas kluster Anda.

- [Prinsipal IAM](#) yang membuat cluster adalah satu-satunya prinsipal yang dapat melakukan panggilan ke server API Kubernetes dengan atau. `kubectl` AWS Management Console Jika Anda ingin prinsipal IAM lainnya memiliki akses ke cluster Anda, maka Anda perlu menambahkannya. Untuk informasi selengkapnya, lihat [the section called “Akses API Kubernetes”](#) dan [the section called “Izin yang diperlukan”](#).
- Sebaiknya blokir akses Pod ke IMDS jika kondisi berikut benar:
 - Anda berencana untuk menetapkan peran IAM ke semua akun layanan Kubernetes Anda sehingga Pod hanya memiliki izin minimum yang mereka butuhkan.
 - Tidak ada Pod dalam kluster yang memerlukan akses ke layanan metadata EC2 instans Amazon (IMDS) karena alasan lain, seperti mengambil Region saat ini. AWS

Untuk informasi selengkapnya, lihat [Membatasi akses ke profil instance yang ditetapkan ke node pekerja](#).

- Konfigurasi Kubernetes [Cluster Autoscaler](#) untuk secara otomatis menyesuaikan jumlah node dalam grup node Anda.
- Menerapkan [aplikasi sampel](#) ke cluster Anda.
- [Atur dan pantau sumber daya kluster](#) dengan alat penting untuk mengelola kluster Anda.

Memperbarui grup node terkelola untuk kluster Anda

Saat Anda memulai pembaruan grup node terkelola, Amazon EKS secara otomatis memperbarui node untuk Anda, menyelesaikan langkah-langkah yang tercantum dalam [Memahami setiap fase pembaruan node](#). Jika Anda menggunakan AMI Amazon EKS yang dioptimalkan, Amazon EKS secara otomatis menerapkan patch keamanan terbaru dan pembaruan sistem operasi ke node Anda sebagai bagian dari versi rilis AMI terbaru.

Ada beberapa skenario yang berguna untuk memperbarui versi atau konfigurasi grup node terkelola Amazon EKS Anda:

- Anda telah memperbarui versi Kubernetes untuk klaster Amazon EKS dan ingin memperbarui simpul Anda untuk menggunakan versi Kubernetes yang sama.
- Versi perilisan AMI yang baru tersedia untuk grup simpul terkelola Anda. Untuk informasi selengkapnya tentang versi AMI, lihat bagian ini:
 - [the section called “Dapatkan informasi versi”](#)
 - [the section called “Bottlerocket”](#)
 - [the section called “Dapatkan informasi versi”](#)
- Anda ingin menyesuaikan jumlah instans minimum, maksimum, atau diinginkan di grup simpul terkelola Anda.
- Anda ingin menambah atau menghapus label Kubernetes dari instans di grup simpul terkelola Anda.
- Anda ingin menambahkan atau menghapus AWS tag dari grup node terkelola Anda.
- Anda harus men-deploy versi templat peluncuran yang baru dengan perubahan konfigurasi, seperti AMI kustom yang diperbarui.
- Anda telah menerapkan add-on Amazon VPC CNI versi 1.9.0 atau yang lebih baru, mengaktifkan add-on untuk delegasi awalan, dan ingin instance Sistem Nitro AWS baru dalam grup node mendukung peningkatan jumlah Pod secara signifikan. Untuk informasi selengkapnya, lihat [the section called “Meningkatkan alamat IP”](#).
- Anda telah mengaktifkan delegasi awalan IP untuk node Windows dan menginginkan instans Sistem AWS Nitro baru dalam grup node untuk mendukung peningkatan jumlah Pod secara signifikan. Untuk informasi selengkapnya, lihat [the section called “Meningkatkan alamat IP”](#).

Jika ada versi rilis AMI yang lebih baru untuk versi Kubernetes grup node terkelola Anda, Anda dapat memperbarui versi grup node Anda untuk menggunakan versi AMI yang lebih baru. Demikian pula, jika klaster Anda menjalankan versi Kubernetes yang lebih baru dari grup node Anda, Anda dapat memperbarui grup node untuk menggunakan versi rilis AMI terbaru agar sesuai dengan versi Kubernetes klaster Anda.

Ketika sebuah node dalam grup node terkelola dihentikan karena operasi penskalaan atau pembaruan, Pod dalam node tersebut akan terkuras terlebih dahulu. Untuk informasi selengkapnya, lihat [the section called “Perbarui detail perilaku”](#).

Memperbarui versi grup simpul

Anda dapat memperbarui versi grup node dengan salah satu dari berikut ini:

- [the section called “eksctl”](#)
- [the section called “AWS Management Console”](#)

Versi yang Anda perbarui tidak boleh lebih besar dari versi pesawat kontrol.

eksctl

Memperbarui grup node terkelola menggunakan **eksctl**

Perbarui grup node terkelola ke rilis AMI terbaru dari versi Kubernetes yang sama yang saat ini digunakan di node dengan perintah berikut. Ganti setiap *example value* dengan nilai-nilai Anda sendiri.

```
eksctl upgrade nodegroup \  
  --name=node-group-name \  
  --cluster=my-cluster \  
  --region=region-code
```

Note

Jika Anda memutakhirkan grup node yang digunakan dengan template peluncuran ke versi template peluncuran baru, tambahkan `--launch-template-version version-number` ke perintah sebelumnya. Template peluncuran harus memenuhi persyaratan yang dijelaskan dalam [Sesuaikan node terkelola dengan templat peluncuran](#). Jika template peluncuran menyertakan AMI kustom, AMI harus memenuhi persyaratan dalam [Menentukan AMI](#). Saat Anda memutakhirkan grup node ke versi template peluncuran yang lebih baru, setiap node didaur ulang agar sesuai dengan konfigurasi baru dari versi template peluncuran yang ditentukan.

Anda tidak dapat langsung memutakhirkan grup node yang digunakan tanpa template peluncuran ke versi template peluncuran baru. Sebaliknya, Anda harus men-deploy grup simpul baru dengan menggunakan templat peluncuran untuk memperbarui grup simpul ke versi templat peluncuran baru.

Anda dapat memutakhirkan grup node ke versi yang sama dengan versi Kubernetes pesawat kontrol. Misalnya, jika Anda memiliki klaster yang menjalankan Kubernetes 1.33, Anda dapat meng-upgrade node yang saat ini menjalankan Kubernetes 1.32 ke versi dengan perintah berikut. 1.33

```
eksctl upgrade nodegroup \  
  --name=node-group-name \  
  --cluster=my-cluster \  
  --region=region-code \  
  --kubernetes-version=1.33
```

AWS Management Console

Memperbarui grup node terkelola menggunakan AWS Management Console

1. Buka [konsol Amazon EKS](#).
2. Pilih klaster yang berisi grup simpul yang akan diperbarui.
3. Jika setidaknya tersedia satu grup simpul yang memiliki pembaruan, maka akan ada kotak yang muncul di bagian atas halaman yang memberitahukan Anda tentang pembaruan yang tersedia. Jika Anda memilih tab Compute, Anda akan melihat Perbarui sekarang di kolom versi rilis AMI di tabel grup Node untuk grup node yang memiliki pembaruan yang tersedia. Untuk memperbarui grup node, pilih Perbarui sekarang.

Anda tidak akan melihat notifikasi untuk grup node yang digunakan dengan AMI kustom. Jika simpul di-deploy dengan AMI kustom, selesaikan langkah-langkah berikut untuk men-deploy AMI kustom yang baru diperbarui.

- a. Buat versi baru AMI Anda.
 - b. Buat versi templat peluncuran baru dengan ID AMI baru.
 - c. Mutakhirkan simpul ke versi templat peluncuran baru.
4. Pada kotak dialog Perbarui versi grup node, aktifkan atau nonaktifkan opsi berikut:
 - Perbarui versi grup node - Opsi ini tidak tersedia jika Anda menerapkan AMI khusus atau AMI Amazon EKS yang dioptimalkan saat ini ada di versi terbaru untuk klaster Anda.
 - Ubah versi template peluncuran - Opsi ini tidak tersedia jika grup simpul dikerahkan tanpa templat peluncuran khusus. Anda hanya dapat memperbarui versi templat peluncuran untuk grup simpul yang telah di-deploy dengan templat peluncuran kustom. Pilih versi Template Luncurkan yang ingin Anda perbarui grup node. Jika grup simpul dikonfigurasi dengan AMI kustom, maka versi yang Anda pilih juga harus menentukan AMI. Saat Anda memutakhirkan

ke versi template peluncuran yang lebih baru, setiap node didaur ulang agar sesuai dengan konfigurasi baru dari versi template peluncuran yang ditentukan.

5. Untuk strategi Update, pilih salah satu opsi berikut:

- Pembaruan bergulir — Opsi ini menghormati anggaran gangguan Pod untuk klaster Anda. Pembaruan gagal jika ada masalah anggaran gangguan Pod yang menyebabkan Amazon EKS tidak dapat menguras Pod yang berjalan di grup node ini dengan baik.
- Update paksa — Opsi ini tidak menghormati anggaran gangguan Pod. Pembaruan terjadi terlepas dari masalah anggaran gangguan Pod dengan memaksa restart node terjadi.

6. Pilih Perbarui.

Edit konfigurasi grup simpul

Anda dapat mengubah beberapa konfigurasi dari grup simpul terkelola.

1. Buka [konsol Amazon EKS](#).
2. Pilih klaster yang berisi grup simpul untuk mengedit.
3. Pilih tab Compute.
4. Pilih grup node yang akan diedit, lalu pilih Edit.
5. (Opsional) Pada halaman grup Edit node, lakukan hal berikut:
 - a. Edit konfigurasi penskalaan grup Node.
 - Ukuran yang diinginkan – Tentukan jumlah simpul saat ini yang harus dipertahankan oleh grup simpul terkelola.
 - Ukuran minimum – Tentukan jumlah simpul minimum yang dapat diskalakan kedalam oleh grup simpul terkelola.
 - Ukuran maksimum – Tentukan jumlah maksimum simpul yang dapat diskalakan keluar oleh grup simpul terkelola. Untuk jumlah maksimal simpul yang didukung dalam grup simpul, lihat [the section called “Kuota layanan”](#).
 - b. (Opsional) Tambahkan atau hapus label Kubernetes ke node di grup node Anda. Label yang ditampilkan di sini hanya label yang telah Anda terapkan dengan Amazon EKS. Label lain mungkin ada di node Anda yang tidak ditampilkan di sini.
 - c. (Opsional) Tambahkan atau hapus taints Kubernetes ke node di grup node Anda. Kecacatan yang ditambahkan dapat memiliki efek, baik **NoSchedule** , **NoExecute** , atau **PreferNoSchedule** . Untuk informasi selengkapnya, lihat [the section called “Node GPU noda”](#).

- d. (Opsional) Tambahkan atau hapus Tag dari sumber daya grup node Anda. Tanda ini hanya diterapkan pada grup simpul Amazon EKS. Mereka tidak menyebar ke sumber daya lain, seperti subnet atau EC2 instance Amazon di grup node.
- e. (Opsional) Edit konfigurasi pembaruan Grup Node. Pilih Angka atau Persentase.
 - Angka — Pilih dan tentukan jumlah node dalam grup node Anda yang dapat diperbarui secara paralel. Node ini tidak akan tersedia selama pembaruan.
 - Persentase — Pilih dan tentukan persentase node dalam grup node Anda yang dapat diperbarui secara paralel. Node ini tidak akan tersedia selama pembaruan. Ini berguna jika Anda memiliki banyak node di grup node Anda.
- f. Setelah selesai mengedit, pilih Simpan perubahan.

Important

Saat memperbarui konfigurasi grup node, memodifikasi [NodegroupScalingConfig](#) tidak menghormati anggaran gangguan Pod (). PDBs Tidak seperti proses [grup node pembaruan](#) (yang menguras node dan respek PDBs selama fase pemutakhiran), memperbarui konfigurasi penskalaan menyebabkan node segera dihentikan melalui panggilan penskalaan Grup Auto Scaling (ASG). Ini terjadi tanpa mempertimbangkan PDBs, terlepas dari ukuran target yang Anda turunkan. Itu berarti ketika Anda mengurangi grup node terkelola Amazon EKS, Pod akan diusir segera setelah node dihentikan, tanpa menghormati apa pun. `desiredSize` PDBs

Memahami setiap fase pembaruan node

Strategi upgrade node pekerja terkelola Amazon EKS memiliki empat fase berbeda yang dijelaskan di bagian berikut.

Fase pengaturan

Fase pengaturan memiliki langkah-langkah berikut:

1. Ini membuat versi template EC2 peluncuran Amazon baru untuk Grup Auto Scaling yang terkait dengan grup node Anda. Versi template peluncuran baru menggunakan AMI target atau versi template peluncuran khusus untuk pembaruan.
2. Ini memperbarui Grup Auto Scaling untuk menggunakan versi template peluncuran terbaru.

3. Ini menentukan jumlah maksimum node untuk meng-upgrade secara paralel menggunakan `updateConfig` properti untuk kelompok node. Maksimum yang tidak tersedia memiliki kuota 100 node. Nilai defaultnya adalah satu node. Untuk informasi selengkapnya, lihat properti [updateConfig](#) di Referensi API Amazon EKS.

Tingkatkan fase

Saat memutakhirkan node dalam grup node terkelola, node yang ditingkatkan diluncurkan di Availability Zone yang sama dengan yang sedang ditingkatkan. Untuk menjamin penempatan ini, kami menggunakan Availability EC2 Zone Rebalancing Amazon. Untuk informasi selengkapnya, lihat [Penyeimbangan Kembali Zona Ketersediaan](#) di Panduan Pengguna Auto EC2 Scaling Amazon. Untuk memenuhi persyaratan ini, ada kemungkinan bahwa kami akan meluncurkan hingga dua instance per Availability Zone di grup node terkelola Anda.

Fase peningkatan skala memiliki langkah-langkah berikut:

1. Ini meningkatkan ukuran maksimum Grup Auto Scaling dan ukuran yang diinginkan dengan ukuran yang lebih besar dari keduanya:
 - Hingga dua kali jumlah Availability Zone yang digunakan oleh Grup Auto Scaling.
 - Upgrade maksimum yang tidak tersedia.

Misalnya, jika grup node Anda memiliki lima Availability Zones dan `maxUnavailable` sebagai satu, proses upgrade dapat meluncurkan maksimal 10 node. Namun ketika `maxUnavailable` 20 (atau apa pun yang lebih tinggi dari 10), proses akan meluncurkan 20 node baru.

2. Setelah menskalakan Grup Auto Scaling, ia memeriksa apakah node yang menggunakan konfigurasi terbaru ada di grup node. Langkah ini hanya berhasil jika memenuhi kriteria ini:
 - Setidaknya satu node baru diluncurkan di setiap Availability Zone di mana node ada.
 - Setiap node baru harus dalam Ready keadaan.
 - Node baru harus memiliki label yang diterapkan Amazon EKS.

Ini adalah label yang diterapkan Amazon EKS pada node pekerja dalam grup simpul biasa:

- `eks.amazonaws.com/nodegroup-image=$amiName`
- `eks.amazonaws.com/nodegroup=$nodeGroupName`

Ini adalah label yang diterapkan Amazon EKS pada node pekerja dalam template peluncuran khusus atau grup node AMI:

- `eks.amazonaws.com/nodegroup-image=$amiName`
- `eks.amazonaws.com/nodegroup=$nodeGroupName`
- `eks.amazonaws.com/sourceLaunchTemplateId=$launchTemplateId`
- `eks.amazonaws.com/sourceLaunchTemplateVersion=$launchTemplateVersion`

3. Ini menandai node sebagai tidak dapat dijadwalkan untuk menghindari penjadwalan Pod baru. Ini juga memberi label node `node.kubernetes.io/exclude-from-external-load-balancers=true` untuk menghapus node lama dari penyeimbang beban sebelum mengakhiri node.

Berikut ini adalah alasan yang diketahui yang menyebabkan `NodeCreationFailure` kesalahan dalam fase ini:

Kapasitas tidak mencukupi di Availability Zone

Ada kemungkinan bahwa Availability Zone mungkin tidak memiliki kapasitas tipe instans yang diminta. Disarankan untuk mengonfigurasi beberapa jenis instance saat membuat grup node terkelola.

EC2 batas instans di akun Anda

Anda mungkin perlu menambah jumlah EC2 instans Amazon yang dapat dijalankan akun Anda secara bersamaan menggunakan Service Quotas. Untuk informasi selengkapnya, lihat [EC2 Service Quotas](#) di Panduan Pengguna Amazon Elastic Compute Cloud untuk Instans Linux.

Data pengguna kustom

Data pengguna khusus terkadang dapat merusak proses bootstrap. Skenario ini dapat menyebabkan `kubenet` tidak dimulai pada node atau node tidak mendapatkan label Amazon EKS yang diharapkan pada mereka. Untuk informasi selengkapnya, lihat [the section called "Menentukan AMI"](#).

Setiap perubahan yang membuat node tidak sehat atau tidak siap

Tekanan disk node, tekanan memori, dan kondisi serupa dapat menyebabkan node tidak akan Ready berstatus.

Setiap node paling bootstrap dalam waktu 15 menit

Jika ada node yang membutuhkan waktu lebih dari 15 menit untuk bootstrap dan bergabung dengan cluster, itu akan menyebabkan pemutakhiran menjadi time out. Ini adalah total runtime

untuk bootstrap node baru yang diukur dari saat node baru diperlukan hingga saat bergabung dengan cluster. Saat memutakhirkan grup node terkelola, penghitung waktu dimulai segera setelah ukuran Grup Auto Scaling meningkat.

Fase upgrade

Fase peningkatan berperilaku dalam dua cara berbeda, tergantung pada strategi pembaruan. Ada dua strategi pembaruan: default dan minimal.

Kami merekomendasikan strategi default di sebagian besar skenario. Ini menciptakan node baru sebelum mengakhiri yang lama, sehingga kapasitas yang tersedia dipertahankan selama fase upgrade. Strategi minimal berguna dalam skenario di mana Anda dibatasi untuk sumber daya atau biaya, misalnya dengan akselerator perangkat keras seperti GPUs. Ini mengakhiri node lama sebelum membuat yang baru, sehingga kapasitas total tidak pernah meningkat melebihi kuantitas yang Anda konfigurasi.

Strategi pembaruan default memiliki langkah-langkah berikut:

1. Ini meningkatkan jumlah node (jumlah yang diinginkan) di Auto Scaling Group, menyebabkan grup node untuk membuat node tambahan.
2. Ini secara acak memilih node yang perlu ditingkatkan, hingga maksimum yang tidak tersedia yang dikonfigurasi untuk grup node.
3. Ini menguras Pod dari node. Jika Pod tidak meninggalkan node dalam waktu 15 menit dan tidak ada force flag, fase upgrade gagal dengan `PodEvictionFailure` error. Untuk skenario ini, Anda dapat menerapkan flag force dengan `update-nodegroup-version` permintaan untuk menghapus Pod.
4. Ini menutup node setelah setiap Pod diusir dan menunggu selama 60 detik. Hal ini dilakukan agar pengontrol layanan tidak mengirim permintaan baru ke node ini dan menghapus node ini dari daftar node aktifnya.
5. Ini mengirimkan permintaan penghentian ke Grup Auto Scaling untuk node yang diapit.
6. Ini mengulangi langkah-langkah upgrade sebelumnya sampai tidak ada node dalam grup node yang digunakan dengan versi sebelumnya dari template peluncuran.

Strategi pembaruan minimal memiliki langkah-langkah berikut:

1. Ini mengikat semua node dari grup node di awal, sehingga pengontrol layanan tidak mengirim permintaan baru ke node ini.

2. Ini secara acak memilih node yang perlu ditingkatkan, hingga maksimum yang tidak tersedia yang dikonfigurasi untuk grup node.
3. Ini menguras Pod dari node yang dipilih. Jika Pod tidak meninggalkan node dalam waktu 15 menit dan tidak ada force flag, fase upgrade gagal dengan PodEvictionFailure error. Untuk skenario ini, Anda dapat menerapkan flag force dengan update-nodegroup-version permintaan untuk menghapus Pod.
4. Setelah setiap Pod diusir dan menunggu selama 60 detik, ia mengirimkan permintaan penghentian ke Grup Auto Scaling untuk node yang dipilih. Grup Auto Scaling membuat node baru (sama dengan jumlah node yang dipilih) untuk menggantikan kapasitas yang hilang.
5. Ini mengulangi langkah-langkah upgrade sebelumnya sampai tidak ada node dalam grup node yang digunakan dengan versi sebelumnya dari template peluncuran.

PodEvictionFailure kesalahan selama fase upgrade

Berikut ini adalah alasan yang diketahui yang menyebabkan PodEvictionFailure kesalahan dalam fase ini:

PDB Agresif

PDB agresif didefinisikan pada Pod atau ada beberapa yang PDBs menunjuk ke Pod yang sama.

Penerapan yang menoleransi semua node

Setelah setiap Pod diusir, node diharapkan kosong karena node [tercemar](#) pada langkah-langkah sebelumnya. Namun, jika penerapan mentolerir setiap taint, maka node lebih cenderung tidak kosong, yang menyebabkan kegagalan pengusuran Pod.

Skala turun fase

Fase penurunan skala mengurangi ukuran maksimum grup Auto Scaling dan ukuran yang diinginkan satu per satu untuk kembali ke nilai sebelum pembaruan dimulai.

Jika alur kerja Upgrade menentukan bahwa Cluster Autoscaler meningkatkan grup node selama fase penurunan skala alur kerja, ia segera keluar tanpa membawa grup node kembali ke ukuran aslinya.

Sesuaikan node terkelola dengan templat peluncuran

Untuk kustomisasi tingkat tertinggi, Anda dapat menerapkan node terkelola menggunakan template peluncuran Anda sendiri. Menggunakan template peluncuran memungkinkan kemampuan seperti berikut:

- Berikan argumen bootstrap pada penerapan node, seperti argumen [kubenet](#) ekstra.
- Tetapkan alamat IP ke Pod dari blok CIDR yang berbeda dari alamat IP yang ditetapkan ke node.
- Terapkan AMI kustom Anda sendiri ke node.
- Terapkan CNI kustom Anda sendiri ke node.

Saat Anda memberikan template peluncuran Anda sendiri saat pertama kali membuat grup node terkelola, Anda juga akan memiliki fleksibilitas yang lebih besar nanti. Selama Anda menerapkan grup node terkelola dengan templat peluncuran Anda sendiri, Anda dapat memperbaruinya secara berulang dengan versi berbeda dari template peluncuran yang sama. Saat Anda memperbarui grup node ke versi template peluncuran yang berbeda, semua node dalam grup didaur ulang agar sesuai dengan konfigurasi baru dari versi template peluncuran yang ditentukan.

Grup node terkelola selalu digunakan dengan template peluncuran untuk digunakan dengan grup Amazon EC2 Auto Scaling. Jika Anda tidak menyediakan template peluncuran, Amazon EKS API membuatnya secara otomatis dengan nilai default di akun Anda. Namun, kami tidak menyarankan Anda memodifikasi templat peluncuran yang dibuat secara otomatis. Selain itu, grup node yang ada yang tidak menggunakan templat peluncuran kustom tidak dapat diperbarui secara langsung. Sebagai gantinya, Anda harus membuat grup node baru dengan template peluncuran khusus untuk melakukannya.

Luncurkan dasar-dasar konfigurasi templat

Anda dapat membuat template peluncuran Amazon EC2 Auto Scaling dengan, AWS Management Console AWS CLI, atau SDK. AWS Untuk selengkapnya, lihat [Membuat Template Peluncuran untuk grup Auto Scaling di Panduan Pengguna Amazon Auto EC2 Scaling](#). Beberapa pengaturan dalam templat peluncuran mirip dengan pengaturan yang digunakan untuk konfigurasi simpul terkelola. Saat menerapkan atau memperbarui grup node dengan template peluncuran, beberapa pengaturan harus ditentukan baik dalam konfigurasi grup node atau template peluncuran. Jangan tentukan pengaturan di kedua tempat. Jika pengaturan ada di tempat yang seharusnya tidak, maka operasi seperti membuat atau memperbarui grup node gagal.

Tabel berikut mencantumkan pengaturan yang dilarang dalam template peluncuran. Ini juga mencantumkan pengaturan serupa, jika ada yang tersedia, yang diperlukan dalam konfigurasi grup node terkelola. Pengaturan yang tercantum adalah pengaturan yang muncul di konsol. Mereka mungkin memiliki nama yang mirip tetapi berbeda di AWS CLI dan SDK.

Peluncuran templat — Dilarang	Konfigurasi grup simpul Amazon EKS
Subnet dalam Antarmuka Jaringan (Tambahkan antarmuka jaringan)	Subnet di bawah konfigurasi jaringan grup Node pada halaman Tentukan jaringan
Profil instans IAM dalam Detail lanjutan	Peran IAM node di bawah konfigurasi grup Node pada halaman grup Configure Node
Perilaku Shutdown dan Perilaku Berhenti - Hibernasi dalam Detail lanjutan. Pertahankan default Jangan sertakan dalam pengaturan template peluncuran di template peluncuran untuk kedua pengaturan.	Tidak setara. Amazon EKS harus mengontrol siklus hidup instans, bukan grup Auto Scaling.

Tabel berikut mencantumkan pengaturan yang dilarang dalam konfigurasi grup node terkelola. Ini juga mencantumkan pengaturan serupa, jika ada yang tersedia, yang diperlukan dalam template peluncuran. Pengaturan yang tercantum adalah pengaturan yang muncul di konsol. Mereka mungkin memiliki nama yang mirip di AWS CLI dan SDK.

Konfigurasi grup simpul Amazon EKS - Dilarang	Luncurkan templat
(Hanya jika Anda menetapkan AMI kustom dalam template peluncuran) Jenis AMI di bawah konfigurasi komputasi grup Node pada halaman konfigurasi Set komputasi dan penskalaan — Tampilan konsol Ditentukan dalam templat peluncuran dan ID AMI yang ditentukan.	Gambar Aplikasi dan OS (Gambar Mesin Amazon) di bawah Meluncurkan konten template - Anda harus menentukan ID jika Anda memiliki salah satu dari persyaratan berikut:
Jika Gambar Aplikasi dan OS (Amazon Machine Image) tidak ditentukan dalam	Menggunakan AMI khusus. Jika Anda menentukan AMI yang tidak memenuhi persyaratan yang tercantum dalam

Konfigurasi grup simpul Amazon EKS - Dilarang	Luncurkan templat
template peluncuran, Anda dapat memilih AMI dalam konfigurasi grup node.	Menentukan AMI, penerapan grup node akan gagal. • Ingin memberikan data pengguna untuk memberikan argumen ke <code>bootstrap.sh</code> file yang disertakan dengan AMI Amazon EKS yang dioptimalkan. Anda dapat mengaktifkan instans untuk menetapkan jumlah alamat IP yang jauh lebih tinggi ke Pod, menetapkan alamat IP ke Pod dari blok CIDR yang berbeda dari instans, atau menerapkan kluster pribadi tanpa akses internet keluar. Untuk informasi selengkapnya, lihat topik berikut. • Tetapkan lebih banyak alamat IP ke node Amazon EKS dengan awalan • Terapkan pod di subnet alternatif dengan jaringan khusus • Menyebarkan kluster pribadi dengan akses internet terbatas • Menentukan AMI
Ukuran disk di bawah konfigurasi komputasi grup Node pada Mengatur halaman konfigurasi komputasi dan penskalaan — Tampilan konsol Ditentukan dalam templat peluncuran.	Ukuran dalam Penyimpanan (Volume) (Tambahkan volume baru). Anda harus menentukan ini dalam templat peluncuran.
SSH key pair di bawah konfigurasi grup Node pada halaman Specify Networking - Konsol menampilkan kunci yang ditentukan dalam template peluncuran atau menampilkan Tidak ditentukan dalam template peluncuran.	Nama pasangan kunci dalam Pasangan kunci (login).

Konfigurasi grup simpul Amazon EKS - Dilarang	Luncurkan templat
Anda tidak dapat menentukan grup keamanan sumber yang diizinkan akses jarak jauh saat menggunakan templat peluncuran.	Grup keamanan dalam Pengaturan jaringan untuk instans atau Grup keamanan di bawah Antarmuka jaringan (Tambahkan antarmuka jaringan), tapi tidak keduanya. Untuk informasi selengkapnya, lihat the section called “Menggunakan grup keamanan kustom” .

Note

- Jika Anda menerapkan grup node menggunakan template peluncuran, tentukan nol atau satu jenis Instance di bawah Meluncurkan konten template dalam template peluncuran. Atau, Anda dapat menentukan 0—20 jenis instans untuk tipe Instance di halaman Setel konfigurasi komputasi dan penskalaan di konsol. Atau, Anda dapat melakukannya menggunakan alat lain yang menggunakan Amazon EKS API. Jika Anda menentukan jenis instance dalam template peluncuran, dan menggunakan template peluncuran tersebut untuk menyebarkan grup node, Anda tidak dapat menentukan jenis instance apa pun di konsol atau menggunakan alat lain yang menggunakan Amazon EKS API. Jika Anda tidak menentukan jenis instance dalam template peluncuran, di konsol, atau menggunakan alat lain yang menggunakan Amazon EKS API, jenis `t3.medium` instance akan digunakan. Jika grup simpul Anda menggunakan tipe kapasitas Spot, sebaiknya tentukan beberapa tipe instans menggunakan konsol. Untuk informasi selengkapnya, lihat [the section called “Tipe kapasitas grup simpul terkelola”](#).
- Jika kontainer apa pun yang Anda terapkan ke grup node menggunakan Layanan Metadata Instans Versi 2, pastikan untuk menetapkan batas hop respons Metadata ke 2 dalam template peluncuran Anda. Untuk informasi selengkapnya, lihat [Metadata instans dan data pengguna](#) di EC2 Panduan Pengguna Amazon. Jika Anda menerapkan grup simpul terkelola tanpa menggunakan templat peluncuran kustom, nilai ini secara otomatis ditetapkan untuk grup simpul dalam templat peluncuran default.
- Template peluncuran tidak mendukung InstanceRequirements fitur yang memungkinkan pemilihan jenis instans yang fleksibel.

Menandai instans Amazon EC2

Anda dapat menggunakan `TagSpecification` parameter template peluncuran untuk menentukan tag mana yang akan diterapkan ke EC2 instance Amazon di grup node Anda. Entitas IAM memanggil `CreateNodegroup` atau `UpdateNodegroupVersion` APIs harus memiliki izin untuk `ec2:RunInstances` dan `ec2:CreateTags`, dan tag harus ditambahkan ke template peluncuran.

Menggunakan grup keamanan kustom

Anda dapat menggunakan template peluncuran untuk menentukan [grup EC2 keamanan](#) Amazon khusus untuk diterapkan ke instance di grup node Anda. Ini dapat berupa parameter grup keamanan tingkat instans atau sebagai bagian dari parameter konfigurasi antarmuka jaringan. Namun, Anda tidak dapat membuat template peluncuran yang menentukan tingkat instans dan grup keamanan antarmuka jaringan. Pertimbangkan kondisi yang berlaku berikut untuk menggunakan grup keamanan kustom dengan grup simpul terkelola:

- Saat menggunakan AWS Management Console, Amazon EKS hanya mengizinkan template peluncuran dengan spesifikasi antarmuka jaringan tunggal.
- Secara default, Amazon EKS menerapkan [grup keamanan klaster](#) untuk instans dalam grup simpul Anda guna memfasilitasi komunikasi antara simpul dan pesawat kontrol. Jika Anda menentukan grup keamanan kustom dalam template peluncuran menggunakan salah satu opsi yang disebutkan sebelumnya, Amazon EKS tidak menambahkan grup keamanan klaster. Jadi, Anda harus memastikan bahwa aturan masuk dan keluar dari grup keamanan Anda memungkinkan komunikasi dengan titik akhir klaster Anda. Jika aturan grup keamanan Anda salah, node pekerja tidak dapat bergabung dengan cluster. Untuk informasi selengkapnya tentang aturan grup keamanan, lihat [the section called “Persyaratan grup keamanan”](#).
- Jika Anda memerlukan akses SSH ke instance di grup node Anda, sertakan grup keamanan yang memungkinkan akses tersebut.

Data EC2 pengguna Amazon

Template peluncuran mencakup bagian untuk data pengguna khusus. Anda dapat menentukan pengaturan konfigurasi untuk grup node Anda di bagian ini tanpa membuat kustom individual secara manual AMIs. Untuk informasi selengkapnya tentang pengaturan yang tersedia untuk Bottlerocket, lihat [Menggunakan](#) data pengguna di. GitHub

Anda dapat menyediakan data EC2 pengguna Amazon di template peluncuran Anda menggunakan `cloud-init` saat meluncurkan instans Anda. Untuk informasi selengkapnya, lihat dokumentasi

[cloud-init](#). Data pengguna Anda dapat digunakan untuk melakukan operasi konfigurasi umum. Ini termasuk operasi berikut:

- [Termasuk pengguna atau grup](#)
- [Menginstal paket](#)

Data EC2 pengguna Amazon dalam template peluncuran yang digunakan dengan grup node terkelola harus dalam format [arsip multi-bagian MIME untuk Amazon Linux AMIs dan format TOMM](#) untuk Bottlerocket. AMIs ini karena data pengguna Anda digabungkan dengan data pengguna Amazon EKS yang diperlukan simpul untuk bergabung dengan kluster. Jangan tentukan perintah apa pun dalam data pengguna Anda yang memulai atau memodifikasi `kubelet`. Ini dilakukan sebagai bagian dari data pengguna yang digabungkan oleh Amazon EKS. Parameter `kubelet` tertentu, seperti pengaturan label pada simpul, dapat dikonfigurasi secara langsung melalui API grup simpul terkelola.

 Note

Untuk informasi selengkapnya tentang `kubelet` kustomisasi lanjutan, termasuk memulainya secara manual atau meneruskan parameter konfigurasi kustom, lihat [the section called “Menentukan AMI”](#). Jika ID AMI kustom ditentukan dalam template peluncuran, Amazon EKS tidak menggabungkan data pengguna.

Rincian berikut memberikan informasi lebih lanjut tentang bagian data pengguna.

Data pengguna Amazon Linux 2

Anda dapat menggabungkan beberapa blok data pengguna menjadi satu file multi-bagian MIME. Misalnya, Anda dapat menggabungkan cloud boot hook yang mengonfigurasi Docker daemon dengan skrip shell data pengguna yang menginstal paket kustom. File multi-bagian MIME terdiri dari komponen berikut:

- Tipe konten dan deklarasi batas bagian – `Content-Type: multipart/mixed; boundary=="MYBOUNDARY=="`
- Deklarasi versi MIME – `MIME-Version: 1.0`
- Satu atau lebih blok data pengguna, yang berisi komponen berikut:
 - Batas pembukaan, yang menandakan awal dari blok data pengguna – `--MYBOUNDARY--`

- Deklarasi tipe konten untuk blok: `Content-Type: text/cloud-config; charset="us-ascii"`. Untuk informasi selengkapnya, lihat dokumentasi [cloud-init](#).
- Isi data pengguna (misalnya, daftar perintah atau `cloud-init` arahan shell).
- Batas penutupan, yang menandakan akhir dari file multi-bagian MIME: `--`
`==MYBOUNDARY==--`

Berikut ini adalah contoh file multi-bagian MIME yang dapat Anda gunakan untuk membuat milik Anda sendiri.

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=="MYBOUNDARY=="

--MYBOUNDARY--
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash
echo "Running custom user data script"

--MYBOUNDARY--
```

Data pengguna Amazon Linux 2023

Amazon Linux 2023 (AL2023) memperkenalkan proses inisialisasi node baru `nodeadm` yang menggunakan skema konfigurasi `YAMG`. Jika Anda menggunakan grup node yang dikelola sendiri atau AMI dengan template peluncuran, Anda sekarang harus menyediakan metadata kluster tambahan secara eksplisit saat membuat grup node baru. [Contoh](#) parameter minimum yang diperlukan adalah sebagai berikut, di `manaapiServerEndpoint`, `certificateAuthority`, dan layanan sekarang `cidr` diperlukan:

```
---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: my-cluster
    apiServerEndpoint: https://example.com
    certificateAuthority: Y2VydGlmawNhdGVBdXRob3JpdHk=
    cidr: 10.100.0.0/16
```

Anda biasanya akan mengatur konfigurasi ini dalam data pengguna Anda, baik apa adanya atau disematkan dalam dokumen multi-bagian MIME:

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="BOUNDARY"

--BOUNDARY
Content-Type: application/node.eks.aws

---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig spec: [...]

--BOUNDARY--
```

Pada tahun AL2, metadata dari parameter ini ditemukan dari panggilan Amazon EKS `DescribeCluster` API. Dengan AL2 023, perilaku ini telah berubah karena panggilan API tambahan berisiko melambat selama peningkatan skala node besar. Perubahan ini tidak memengaruhi Anda jika Anda menggunakan grup node terkelola tanpa templat peluncuran atau jika Anda menggunakan Karpenter. Untuk informasi selengkapnya tentang `certificateAuthority` dan `layancidr`, lihat [DescribeCluster](#) di Referensi API Amazon EKS.

Berikut adalah contoh lengkap data pengguna AL2 023 yang menggabungkan skrip shell untuk menyesuaikan node (seperti menginstal paket atau gambar wadah pra-caching) dengan konfigurasi yang diperlukan. `nodeadm` Contoh ini menunjukkan kustomisasi umum termasuk: *

- * Menginstal paket sistem tambahan
- * Gambar kontainer pra-caching untuk meningkatkan waktu startup Pod
- * Menyiapkan konfigurasi proxy HTTP
- * Mengkonfigurasi flag untuk pelabelan node

`kubelet`

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="BOUNDARY"

--BOUNDARY
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash
set -o errexit
set -o pipefail
set -o nounset
```

```
# Install additional packages
yum install -y httpd jq iptables-services

# Pre-cache commonly used container images
nohup docker pull public.ecr.aws/eks-distro/kubernetes/pause:3.2 &

# Configure HTTP proxy if needed
cat > /etc/profile.d/http-proxy.sh << 'EOF'
export HTTP_PROXY="http://proxy.example.com:3128"
export HTTPS_PROXY="http://proxy.example.com:3128"
export NO_PROXY="localhost,127.0.0.1,169.254.169.254,.internal"
EOF

--BOUNDARY
Content-Type: application/node.eks.aws

apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: my-cluster
    apiServerEndpoint: https://example.com
    certificateAuthority: Y2VyZGlmaWNhdGVBdXRob3JpdHk=
    cidr: 10.100.0.0/16
  kubelet:
    config:
      clusterDNS:
        - 10.100.0.10
    flags:
      - --node-labels=app=my-app,environment=production

--BOUNDARY--
```

Data pengguna bottlerocket

Bottlerocket menyusun data pengguna dalam format TOMB. Anda dapat memberikan data pengguna untuk digabungkan dengan data pengguna yang disediakan oleh Amazon EKS. Misalnya, Anda dapat memberikan kubelet pengaturan tambahan.

```
[settings.kubernetes.system-reserved]
cpu = "10m"
memory = "100Mi"
ephemeral-storage= "1Gi"
```

Untuk informasi selengkapnya tentang pengaturan yang didukung, lihat dokumentasi [Bottlerocket](#). Anda dapat mengonfigurasi label node dan [taints](#) dalam data pengguna Anda. Namun, kami menyarankan Anda mengonfigurasi ini dalam grup node Anda sebagai gantinya. Amazon EKS menerapkan konfigurasi ini saat Anda melakukannya.

Saat data pengguna digabungkan, pemformatan tidak dipertahankan, tetapi kontennya tetap sama. Konfigurasi yang Anda berikan dalam data pengguna akan mengesampingkan pengaturan apa pun yang dikonfigurasi oleh Amazon EKS. Jadi, jika Anda menetapkan `settings.kubernetes.max-pods` atau `settings.kubernetes.cluster-dns-ip`, nilai-nilai ini dalam data pengguna Anda diterapkan ke node.

Amazon EKS tidak mendukung semua TOMB yang valid. Berikut ini adalah daftar format yang tidak didukung yang diketahui:

- Kutipan dalam kunci yang dikutip: `'quoted "value"' = "value"`
- Kutipan lolos dalam nilai: `str = "I'm a string. \"You can quote me\""`
- Campuran pelampung dan bilangan bulat: `numbers = [ 0.1, 0.2, 0.5, 1, 2, 5 ]`
- Jenis campuran dalam array: `contributors = [ "foo@example.com", { name = "Baz", email = "baz@example.com" } ]`
- Header bertanda kurung dengan tombol yang dikutip: `[foo."bar.baz"]`

Data pengguna Windows

Data pengguna Windows menggunakan PowerShell perintah. Saat membuat grup node terkelola, data pengguna kustom Anda digabungkan dengan data pengguna terkelola Amazon EKS. PowerShell Perintah Anda didahulukan, diikuti oleh perintah data pengguna terkelola, semuanya dalam satu `<powershell></powershell>` tag.

Important

Saat membuat grup node Windows, Amazon EKS memperbarui `aws-auth ConfigMap` untuk memungkinkan node berbasis Linux bergabung dengan cluster. Layanan tidak secara otomatis mengkonfigurasi izin untuk Windows AMLs. Jika Anda menggunakan node Windows, Anda harus mengelola akses baik melalui API entri akses atau dengan memperbarui secara `aws-auth ConfigMap` langsung. Untuk informasi selengkapnya, lihat [the section called “Aktifkan dukungan Windows”](#).

Note

Ketika tidak ada ID AMI yang ditentukan dalam template peluncuran, jangan gunakan skrip Bootstrap Windows Amazon EKS dalam data pengguna untuk mengonfigurasi Amazon EKS.

Contoh data pengguna adalah sebagai berikut.

```
<powershell>
Write-Host "Running custom user data script"
</powershell>
```

Menentukan AMI

Jika Anda memiliki salah satu dari persyaratan berikut, maka tentukan ID AMI di ImageId bidang template peluncuran Anda. Pilih persyaratan yang Anda miliki untuk informasi tambahan.

Berikan data pengguna untuk meneruskan argumen ke **bootstrap.sh** file yang disertakan dengan AMI Linux/Bottlerocket Amazon EKS yang dioptimalkan

Bootstrapping adalah istilah yang digunakan untuk menggambarkan penambahan perintah yang dapat dijalankan ketika sebuah instance dimulai. [Misalnya, bootstrap memungkinkan penggunaan argumen kubelet ekstra](#). Anda dapat meneruskan argumen ke bootstrap.sh skrip dengan menggunakan eksctl tanpa menentukan template peluncuran. Atau Anda dapat melakukannya dengan menentukan informasi di bagian data pengguna dari template peluncuran.

eksctl tanpa menentukan template peluncuran

Buat file bernama *my-nodegroup.yaml* dengan isi berikut ini. Ganti setiap *example value* dengan nilai-nilai Anda sendiri. `--dns-cluster-ip` Argumen `--apiserver-endpoint` `--b64-cluster-ca`, dan bersifat opsional. Namun, mendefinisikannya memungkinkan bootstrap.sh skrip untuk menghindari `describeCluster` panggilan. Ini berguna dalam pengaturan cluster pribadi atau cluster tempat Anda sering menskalakan node masuk dan keluar. Untuk informasi selengkapnya tentang bootstrap.sh skrip, lihat file [bootstrap.sh](#) di GitHub.

- Satu-satunya argumen yang diperlukan adalah nama cluster (*my-cluster*).
- Untuk mengambil ID AMI yang dioptimalkan `ami-1234567890abcdef0`, lihat bagian berikut:

- [Ambil AMI Amazon Linux yang direkomendasikan IDs](#)
- [Ambil Bottlerocket AMI yang direkomendasikan IDs](#)
- [Ambil Microsoft Windows AMI yang direkomendasikan IDs](#)
- Untuk mengambil *certificate-authority* untuk cluster Anda, jalankan perintah berikut.

```
aws eks describe-cluster --query "cluster.certificateAuthority.data" --output text
--name my-cluster --region region-code
```

- Untuk mengambil *api-server-endpoint* untuk cluster Anda, jalankan perintah berikut.

```
aws eks describe-cluster --query "cluster.endpoint" --output text --name my-
cluster --region region-code
```

- Nilai untuk `--dns-cluster-ip` adalah CIDR layanan Anda dengan `.10` di akhir. Untuk mengambil *service-cidr* untuk cluster Anda, jalankan perintah berikut. Misalnya, jika nilai yang dikembalikan adalah `ipv4 10.100.0.0/16`, maka nilai Anda adalah `10.100.0.10`.

```
aws eks describe-cluster --query "cluster.kubernetesNetworkConfig.serviceIpv4Cidr"
--output text --name my-cluster --region region-code
```

- Contoh ini memberikan kubelet argumen untuk menetapkan `max-pods` nilai kustom menggunakan `bootstrap.sh` skrip yang disertakan dengan AMI Amazon EKS yang dioptimalkan. Nama grup node tidak boleh lebih dari 63 karakter. Itu harus dimulai dengan huruf atau digit, tetapi juga dapat menyertakan tanda hubung dan garis bawah untuk karakter yang tersisa. Untuk bantuan dalam memilih *my-max-pods-value*, lihat [the section called "Amazon EKS merekomendasikan Pod maksimum untuk setiap jenis EC2 instans Amazon"](#).

```
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code

managedNodeGroups:
- name: my-nodegroup
  ami: ami-1234567890abcdef0
  instanceType: m5.large
  privateNetworking: true
```

```

disableIMDSv1: true
labels: { x86-al2-specified-mng }
overrideBootstrapCommand: |
  #!/bin/bash
  /etc/eks/bootstrap.sh my-cluster \
    --b64-cluster-ca certificate-authority \
    --apiserver-endpoint api-server-endpoint \
    --dns-cluster-ip service-cidr.10 \
    --kubenet-extra-args '--max-pods=my-max-pods-value' \
    --use-max-pods false

```

Untuk setiap opsi eksctl config file yang tersedia, lihat [Skema file Config dalam dokumentasi](#). eksctl eksctlUtilitas masih membuat template peluncuran untuk Anda dan mengisi data penggunaannya dengan data yang Anda berikan dalam config file.

Buat grup node dengan perintah berikut.

```
eksctl create nodegroup --config-file=my-nodegroup.yaml
```

Data pengguna dalam template peluncuran

Tentukan informasi berikut di bagian data pengguna dari template peluncuran Anda. Ganti setiap *example value* dengan nilai-nilai Anda sendiri. `--dns-cluster-ip` Argumen `--apiserver-endpoint` `--b64-cluster-ca`, dan bersifat opsional. Namun, mendefinisikannya memungkinkan `bootstrap.sh` skrip untuk menghindari `describeCluster` panggilan. Ini berguna dalam pengaturan cluster pribadi atau cluster tempat Anda sering menskalakan node masuk dan keluar. Untuk informasi selengkapnya tentang `bootstrap.sh` skrip, lihat file [bootstrap.sh](#) di GitHub.

- Satu-satunya argumen yang diperlukan adalah nama cluster (*my-cluster*).
- Untuk mengambil *certificate-authority* untuk cluster Anda, jalankan perintah berikut.

```
aws eks describe-cluster --query "cluster.certificateAuthority.data" --output text
--name my-cluster --region region-code
```

- Untuk mengambil *api-server-endpoint* untuk cluster Anda, jalankan perintah berikut.

```
aws eks describe-cluster --query "cluster.endpoint" --output text --name my-
cluster --region region-code
```

- Nilai untuk `--dns-cluster-ip` adalah CIDR layanan Anda dengan `.10` di akhir. Untuk mengambil `service-cidr` untuk cluster Anda, jalankan perintah berikut. Misalnya, jika nilai yang dikembalikan adalah `10.100.0.0/16`, maka nilai Anda adalah `10.100.0.10`.

```
aws eks describe-cluster --query "cluster.kubernetesNetworkConfig.serviceIpv4Cidr"
--output text --name my-cluster --region region-code
```

- Contoh ini memberikan kubelet argumen untuk menetapkan `max-pods` nilai kustom menggunakan `bootstrap.sh` skrip yang disertakan dengan AMI Amazon EKS yang dioptimalkan. Untuk bantuan dalam memilih `my-max-pods-value`, lihat [the section called "Amazon EKS merekomendasikan Pod maksimum untuk setiap jenis EC2 instans Amazon"](#).

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=="MYBOUNDARY=="

--MYBOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash
set -ex
/etc/eks/bootstrap.sh my-cluster \
  --b64-cluster-ca certificate-authority \
  --apiserver-endpoint api-server-endpoint \
  --dns-cluster-ip service-cidr.10 \
  --kubelet-extra-args '--max-pods=my-max-pods-value' \
  --use-max-pods false

--MYBOUNDARY==
```

Berikan data pengguna untuk meneruskan argumen ke **Start-EKSBootstrap.ps1** file yang disertakan dengan AMI Windows Amazon EKS yang dioptimalkan

Bootstrapping adalah istilah yang digunakan untuk menggambarkan penambahan perintah yang dapat dijalankan ketika sebuah instance dimulai. Anda dapat meneruskan argumen ke **Start-EKSBootstrap.ps1** skrip dengan menggunakan `eksctl` tanpa menentukan template peluncuran. Atau Anda dapat melakukannya dengan menentukan informasi di bagian data pengguna dari template peluncuran.

Jika Anda ingin menentukan ID AMI Windows kustom, ingatlah pertimbangan berikut:

- Anda harus menggunakan template peluncuran dan memberikan perintah bootstrap yang diperlukan di bagian data pengguna. Untuk mengambil ID Windows yang Anda inginkan, Anda dapat menggunakan tabel di [Buat node dengan Windows AMIs yang dioptimalkan](#).
- Ada beberapa batasan dan kondisi. Misalnya, Anda harus menambahkan `eks:kube-proxy-windows` ke peta konfigurasi AWS IAM Authenticator Anda. Untuk informasi selengkapnya, lihat [the section called “Batas dan ketentuan saat menentukan ID AMI”](#).

Tentukan informasi berikut di bagian data pengguna dari template peluncuran Anda. Ganti setiap *example value* dengan nilai-nilai Anda sendiri. `-DNSClusterIPArgumen - APIServerEndpoint-Base64ClusterCA`, dan bersifat opsional. Namun, mendefinisikannya memungkinkan `Start-EKSBootstrap.ps1` skrip untuk menghindari `describeCluster` panggilan.

- Satu-satunya argumen yang diperlukan adalah nama cluster (*my-cluster*).
- Untuk mengambil *certificate-authority* untuk cluster Anda, jalankan perintah berikut.

```
aws eks describe-cluster --query "cluster.certificateAuthority.data" --output text --name my-cluster --region region-code
```

- Untuk mengambil *api-server-endpoint* untuk cluster Anda, jalankan perintah berikut.

```
aws eks describe-cluster --query "cluster.endpoint" --output text --name my-cluster --region region-code
```

- Nilai untuk `--dns-cluster-ip` adalah CIDR layanan Anda dengan `.10` di akhir. Untuk mengambil *service-cidr* untuk cluster Anda, jalankan perintah berikut. Misalnya, jika nilai yang dikembalikan adalah `ipv4 10.100.0.0/16`, maka nilai Anda adalah *10.100.0.10*.

```
aws eks describe-cluster --query "cluster.kubernetesNetworkConfig.serviceIpv4Cidr" --output text --name my-cluster --region region-code
```

- Untuk argumen tambahan, lihat [the section called “Parameter konfigurasi skrip bootstrap”](#).

Note

Jika Anda menggunakan layanan khusus CIDR, maka Anda perlu menentukannya menggunakan `-ServiceCIDR` parameter. Jika tidak, resolusi DNS untuk Pod di cluster akan gagal.

```
<powershell>
[string]$EKSBootstrapScriptFile = "$env:ProgramFiles\Amazon\EKS\Start-EKSBootstrap.ps1"
& $EKSBootstrapScriptFile -EKSClusterName my-cluster `
    -Base64ClusterCA certificate-authority `
    -APIServerEndpoint api-server-endpoint `
    -DNSClusterIP service-cidr.10
</powershell>
```

Jalankan AMI khusus karena persyaratan keamanan, kepatuhan, atau kebijakan internal tertentu

Untuk informasi selengkapnya, lihat [Gambar Mesin Amazon \(AMI\)](#) di Panduan EC2 Pengguna Amazon. Spesifikasi build Amazon EKS AMI berisi sumber daya dan skrip konfigurasi untuk membuat Amazon EKS AMI khusus berdasarkan Amazon Linux. Untuk informasi selengkapnya, lihat [Amazon EKS AMI Build Specification](#) on GitHub. Untuk membuat kustom yang AMIs diinstal dengan sistem operasi lain, lihat [Amazon EKS Sample Custom AMIs](#) di GitHub.

Anda tidak dapat menggunakan referensi parameter dinamis untuk AMI IDs di Template Peluncuran yang digunakan dengan grup node terkelola.

Important

Saat menentukan AMI, Amazon EKS tidak menggabungkan data pengguna apa pun. Sebaliknya, Anda bertanggung jawab untuk menyediakan bootstrap perintah yang diperlukan untuk node untuk bergabung dengan cluster. Jika simpul Anda gagal untuk bergabung dengan klaster, tindakan Amazon EKS `CreateNodegroup` dan `UpdateNodegroupVersion` juga gagal.

Batas dan ketentuan saat menentukan ID AMI

Berikut ini adalah batasan dan kondisi yang terkait dengan menentukan ID AMI dengan grup node terkelola:

- Anda harus membuat grup node baru untuk beralih antara menentukan ID AMI dalam template peluncuran dan tidak menentukan ID AMI.
- Anda tidak diberi tahu di konsol saat versi AMI yang lebih baru tersedia. Untuk memperbarui grup node ke versi AMI yang lebih baru, Anda perlu membuat versi baru template peluncuran dengan ID AMI yang diperbarui. Kemudian, Anda perlu memperbarui grup node dengan versi template peluncuran baru.

- Bidang berikut tidak dapat disetel di API jika Anda menentukan ID AMI:
 - `amiType`
 - `releaseVersion`
 - `version`
- Setiap taints set dalam API diterapkan secara asinkron jika Anda menentukan ID AMI. Untuk menerapkan taints sebelum node bergabung dengan cluster, Anda harus meneruskan taints ke kubelet dalam data pengguna Anda menggunakan flag baris `--register-with-taints` perintah. Untuk informasi selengkapnya, lihat [kubelet](#) dalam dokumentasi Kubernetes.
- Saat menentukan ID AMI khusus untuk grup node terkelola Windows, tambahkan `eks:kube-proxy-windows` ke peta konfigurasi AWS IAM Authenticator Anda. Ini diperlukan agar DNS berfungsi dengan baik.
 - a. Buka peta konfigurasi AWS IAM Authenticator untuk mengedit.

```
kubectl edit -n kube-system cm aws-auth
```

- b. Tambahkan entri ini ke groups daftar di bawah masing-masing rolearn yang terkait dengan node Windows. Peta konfigurasi Anda akan terlihat mirip [aws-auth-cm-windowsdengan.yaml](#).

```
- eks:kube-proxy-windows
```

- c. Simpan file, dan tutup editor teks Anda.

Menghapus grup node terkelola dari klaster

Topik ini menjelaskan cara menghapus grup simpul terkelola Amazon EKS. Saat Anda menghapus grup node terkelola, Amazon EKS pertama-tama menetapkan ukuran minimum, maksimum, dan yang diinginkan dari grup Auto Scaling Anda ke nol. Ini kemudian menyebabkan grup node Anda menurunkan skala.

Sebelum setiap instance dihentikan, Amazon EKS mengirimkan sinyal untuk menguras Pod dari node tersebut. Jika Pod tidak terkuras setelah beberapa menit, Amazon EKS memungkinkan Auto Scaling melanjutkan penghentian instance. Setelah setiap instance dihentikan, grup Auto Scaling akan dihapus.

Important

Jika Anda menghapus grup node terkelola yang menggunakan peran IAM node yang tidak digunakan oleh grup node terkelola lainnya di kluster, peran tersebut akan dihapus dari `aws-auth ConfigMap`. Jika salah satu grup node yang dikelola sendiri di cluster menggunakan peran IAM node yang sama, node yang dikelola sendiri pindah ke status `NotReady`. Selain itu, operasi cluster juga terganggu. Untuk menambahkan pemetaan untuk peran yang Anda gunakan hanya untuk grup node yang dikelola sendiri, lihat [the section called “Buat entri akses”](#), apakah versi platform kluster Anda setidaknya merupakan versi minimum yang tercantum di bagian prasyarat [Grant IAM pengguna akses ke Kubernetes dengan entri akses EKS](#). Jika versi platform Anda lebih awal dari versi minimum yang diperlukan untuk entri akses, Anda dapat menambahkan entri kembali ke `aws-authConfigMap`. Untuk informasi lebih lanjut, masukkan `eksctl create iamidentitymapping --help` di terminal Anda.

Anda dapat menghapus grup node terkelola dengan:

- [the section called “eksctl”](#)
- [the section called “AWS Management Console”](#)
- [the section called “AWS CLI”](#)

eksctl

Hapus grup node terkelola dengan **eksctl**

Masukkan perintah berikut. Ganti setiap *example value* dengan nilai-nilai Anda sendiri.

```
eksctl delete nodegroup \  
  --cluster my-cluster \  
  --name my-mng \  
  --region region-code
```

Untuk opsi lainnya, lihat [Menghapus dan menguras nodegroup](#) dalam dokumentasi `eksctl`

AWS Management Console

Hapus grup node terkelola dengan AWS Management Console

1. Buka [konsol Amazon EKS](#).

2. Pada halaman Clusters, pilih cluster yang berisi grup node untuk dihapus.
3. Pada halaman cluster yang dipilih, pilih tab Compute.
4. Di bagian Node groups, pilih grup node yang akan dihapus. Lalu pilih Hapus.
5. Dalam kotak dialog Hapus konfirmasi grup simpul, masukkan nama grup simpul. Lalu pilih Hapus.

AWS CLI

Hapus grup node dikelola dengan AWS CLI

1. Masukkan perintah berikut. Ganti setiap *example value* dengan nilai-nilai Anda sendiri.

```
aws eks delete-nodegroup \  
  --cluster-name my-cluster \  
  --nodegroup-name my-mng \  
  --region region-code
```

2. Gunakan tombol panah pada keyboard Anda untuk menggulir melalui output respons. Tekan q tombol ketika Anda selesai.

Untuk opsi lainnya, lihat [delete-nodegroup](#) perintah di AWS CLI Command Reference.

Pertahankan node sendiri dengan node yang dikelola sendiri

Sebuah klaster berisi satu atau beberapa EC2 node Amazon tempat Pod dijadwalkan. Node Amazon EKS berjalan di AWS akun Anda dan terhubung ke bidang kontrol klaster Anda melalui titik akhir server API cluster. Anda ditagih untuk mereka berdasarkan EC2 harga Amazon. Untuk informasi selengkapnya, lihat [EC2 harga Amazon](#).

Sebuah klaster dapat berisi beberapa grup simpul. Setiap grup node berisi satu atau beberapa node yang digunakan dalam grup [Amazon EC2 Auto Scaling](#). [Jenis instance node dalam grup dapat bervariasi, seperti saat menggunakan pemilihan tipe instance berbasis atribut dengan Karpenter](#). Semua instance dalam grup node harus menggunakan peran [IAM node Amazon EKS](#).

Amazon EKS menyediakan Gambar Mesin Amazon khusus (AMIs) yang disebut Amazon EKS dioptimalkan AMIs. Itu AMIs dikonfigurasi untuk bekerja dengan Amazon EKS. Komponen mereka termasuk `containerd`, `kubelet`, dan AWS IAM Authenticator. Ini AMIs juga berisi [skrip bootstrap](#) khusus yang memungkinkannya menemukan dan terhubung ke bidang kontrol cluster Anda secara otomatis.

Jika Anda membatasi akses ke titik akhir publik kluster Anda menggunakan blok CIDR, sebaiknya Anda juga mengaktifkan akses titik akhir pribadi. Ini agar node dapat berkomunikasi dengan cluster. Jika titik akhir privat diaktifkan, blok CIDR yang Anda tentukan untuk akses publik harus menyertakan sumber jalan keluar dari VPC Anda. Untuk informasi selengkapnya, lihat [the section called “Akses titik akhir kluster”](#).

Untuk menambahkan simpul-simpul yang dikelola sendiri untuk kluster Amazon EKS Anda, lihat topik-topik berikutnya. Jika Anda meluncurkan node yang dikelola sendiri secara manual, tambahkan tag berikut ke setiap node sambil memastikan bahwa itu `<cluster-name>` cocok dengan cluster Anda. Untuk informasi selengkapnya, lihat [Penambahan dan penghapusan tanda pada sumber daya individu](#). Jika Anda mengikuti langkah-langkah dalam panduan berikut, tag yang diperlukan secara otomatis ditambahkan ke node untuk Anda.

Kunci	Nilai
<code>kubernetes.io/cluster/<cluster-name></code>	<code>owned</code>

Important

Tag di Amazon EC2 Instance Metadata Service (IMDS) tidak kompatibel dengan node EKS. Saat Tag Metadata Instance diaktifkan, penggunaan garis miring maju (') dalam nilai tag dicegah. Keterbatasan ini dapat menyebabkan kegagalan peluncuran instance, terutama saat menggunakan alat manajemen node seperti Karpenter atau Cluster Autoscaler, karena layanan ini mengandalkan tag yang berisi garis miring ke depan untuk fungsionalitas yang tepat.

Untuk informasi selengkapnya terkait simpul dari perspektif Kubernetes umum, lihat [Simpul](#) dalam dokumentasi Kubernetes.

Topik

- [Buat node Amazon Linux yang dikelola sendiri](#)
- [Buat node Bottlerocket yang dikelola sendiri](#)
- [Buat node Microsoft Windows yang dikelola sendiri](#)
- [Buat node Linux Ubuntu yang dikelola sendiri](#)

- [Perbarui node yang dikelola sendiri untuk kluster Anda](#)

Buat node Amazon Linux yang dikelola sendiri

Topik ini menjelaskan bagaimana Anda dapat meluncurkan grup Auto Scaling dari node Linux yang mendaftar dengan kluster Amazon EKS Anda. Setelah simpul bergabung dengan kluster, Anda dapat men-deploy aplikasi Kubernetes pada mereka. Anda juga dapat meluncurkan node Amazon Linux yang dikelola sendiri dengan `eksctl` atau AWS Management Console Jika Anda perlu meluncurkan node di AWS Outposts, lihat [the section called “Simpul”](#)

- Sebuah kluster Amazon EKS yang sudah ada. Untuk menyebarkan satu, lihat [the section called “Membuat kluster”](#). Jika Anda memiliki subnet di AWS Wilayah tempat Anda mengaktifkan AWS Outposts, AWS Wavelength, atau AWS Local Zones, subnet tersebut tidak boleh diteruskan saat Anda membuat kluster.
- Peran IAM yang ada untuk digunakan node. Untuk membuatnya, lihat [the section called “IAM role simpul”](#). Jika peran ini tidak memiliki salah satu kebijakan untuk VPC CNI, peran terpisah yang mengikuti diperlukan untuk pod VPC CNI.
- (Opsional, tetapi disarankan) Plugin Amazon VPC CNI untuk add-on Kubernetes dikonfigurasi dengan peran IAM sendiri yang memiliki kebijakan IAM yang diperlukan yang melekat padanya. Untuk informasi selengkapnya, lihat [the section called “Konfigurasi untuk IRSA”](#).
- Keakraban dengan pertimbangan yang tercantum dalam [Pilih jenis instans EC2 node Amazon yang](#) optimal. Bergantung pada jenis instans yang Anda pilih, mungkin ada prasyarat tambahan untuk cluster dan VPC Anda.

Anda dapat meluncurkan node Linux yang dikelola sendiri menggunakan salah satu dari berikut ini:

- [the section called “eksctl”](#)
- [the section called “AWS Management Console”](#)

eksctl

Luncurkan node Linux yang dikelola sendiri menggunakan **eksctl**

1. Instal versi `0.210.0` atau yang lebih baru dari alat baris perintah yang diinstal pada perangkat Anda atau AWS CloudShell. Untuk menginstal atau memperbaruieksctl, lihat [Instalasi](#) dalam eksctl dokumentasi.

2. (Opsional) Jika kebijakan IAM terkelola `AmazonEKS_CNI_Policy` dilampirkan ke peran IAM [node Amazon EKS Anda](#), kami sarankan untuk menetakannya ke peran IAM yang Anda kaitkan ke akun layanan Kubernetes sebagai gantinya. `aws-node` Untuk informasi selengkapnya, lihat [the section called “Konfigurasi untuk IRSA”](#).
3. Perintah berikutnya membuat grup simpul dalam kluster yang ada. Ganti `al-nodes` dengan nama untuk grup node Anda. Nama grup node tidak boleh lebih dari 63 karakter. Itu harus dimulai dengan huruf atau digit, tetapi juga dapat menyertakan tanda hubung dan garis bawah untuk karakter yang tersisa. Ganti `my-cluster` dengan nama kluster Anda. Nama hanya dapat berisi karakter alfanumerik (peka huruf besar/kecil) dan tanda hubung. Itu harus dimulai dengan karakter alfanumerik dan tidak boleh lebih dari 100 karakter. Nama harus unik di dalam AWS Wilayah dan AWS akun tempat Anda membuat kluster. Ganti sisanya `example value` dengan nilai Anda sendiri. Simpul dibuat dengan versi Kubernetes yang sama dengan bidang kendali, secara default.

Sebelum memilih nilai untuk `--node-type`, tinjau [Pilih jenis instans EC2 node Amazon yang optimal](#).

Ganti `my-key` dengan nama Amazon EC2 key pair atau public key Anda. Kunci ini digunakan untuk SSH ke simpul Anda setelah diluncurkan. Jika Anda belum memiliki EC2 key pair Amazon, Anda dapat membuatnya di AWS Management Console. Untuk informasi selengkapnya, lihat [pasangan EC2 kunci Amazon](#) di Panduan EC2 Pengguna Amazon.

Buat grup simpul Anda dengan perintah berikut.

Important

Jika Anda ingin menyebarkan grup node ke subnet AWS Outposts, Wavelength, atau Local Zone, ada pertimbangan tambahan:

- Subnet tidak boleh diteruskan saat Anda membuat cluster.
- Anda harus membuat grup node dengan file konfigurasi yang menentukan subnet dan `volumeType`: `gp2` Untuk informasi selengkapnya, lihat [Membuat nodegroup dari file konfigurasi dan skema file Config](#) dalam dokumentasi. `eksctl`

```
eksctl create nodegroup \  
  --cluster my-cluster \  
  --name al-nodes \  
  --node-type t3.medium \  
  --nodes 3 \  

```



```
--nodes-min 1 \  
--nodes-max 4 \  
--ssh-access \  
--managed=false \  
--ssh-public-key my-key
```

Untuk menyebarkan grup simpul yang:

- dapat menetapkan jumlah alamat IP yang jauh lebih tinggi ke Pod daripada konfigurasi default, lihat [the section called “Meningkatkan alamat IP”](#).
- dapat menetapkan IPv4 alamat ke Pod dari blok CIDR yang berbeda dari instance, lihat. [the section called “Jaringan kustom”](#)
- dapat menetapkan IPv6 alamat ke Pod dan layanan, lihat [the section called “IPv6”](#).
- tidak memiliki akses internet keluar, lihat [the section called “klaster privat”](#).

Untuk daftar lengkap semua opsi dan default yang tersedia, masukkan perintah berikut.

```
eksctl create nodegroup --help
```

Jika node gagal bergabung dengan cluster, maka lihat [the section called “Simpul gagal untuk bergabung dengan klaster”](#) di chapter Troubleshooting.

Contoh output adalah sebagai berikut. Beberapa baris adalah output sementara node dibuat. Salah satu baris terakhir dari output adalah baris contoh berikutnya.

```
[#] created 1 nodegroup(s) in cluster "my-cluster"
```

4. (Optional) Deploy [aplikasi sampel](#) untuk menguji simpul klaster dan Linux Anda.
5. Sebaiknya blokir akses Pod ke IMDS jika kondisi berikut benar:
 - Anda berencana untuk menetapkan peran IAM ke semua akun layanan Kubernetes Anda sehingga Pod hanya memiliki izin minimum yang mereka butuhkan.
 - Tidak ada Pod dalam klaster yang memerlukan akses ke layanan metadata EC2 instans Amazon (IMDS) karena alasan lain, seperti mengambil Region saat ini. AWS

Untuk informasi selengkapnya, lihat [Membatasi akses ke profil instance yang ditetapkan ke node pekerja](#).

AWS Management Console

Langkah 1: Luncurkan node Linux yang dikelola sendiri menggunakan AWS Management Console

1. Unduh versi terbaru dari AWS CloudFormation template.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2022-12-23/amazon-eks-nodegroup.yaml
```

2. Tunggu status klaster Anda ditampilkan sebagai ACTIVE. Jika Anda meluncurkan node Anda sebelum cluster aktif, node gagal mendaftar dengan cluster dan Anda harus meluncurkannya kembali.
3. Buka [konsol AWS CloudFormation](#).
4. Pilih Buat tumpukan dan kemudian pilih Dengan sumber daya baru (standar).
5. Untuk Menentukan templat, pilih Unggah sebuah file templat dan kemudian pilih Pilih file.
6. Pilih `amazon-eks-nodegroup.yaml` file yang Anda unduh.
7. Pilih Selanjutnya.
8. Pada halaman Tentukan detail tumpukan, masukkan parameter berikut yang sesuai, lalu pilih Berikutnya:
 - Nama tumpukan: Pilih nama tumpukan untuk AWS CloudFormation tumpukan Anda. Misalnya, Anda bisa menyebutnya `my-cluster-nodes`. Nama hanya dapat berisi karakter alfanumerik (peka huruf besar/kecil) dan tanda hubung. Itu harus dimulai dengan karakter alfanumerik dan tidak boleh lebih dari 100 karakter. Nama harus unik di dalam AWS Wilayah dan AWS akun tempat Anda membuat klaster.
 - ClusterName: Masukkan nama yang Anda gunakan saat membuat cluster Amazon EKS Anda. Nama ini harus sama dengan nama cluster atau node Anda tidak dapat bergabung dengan cluster.
 - ClusterControlPlaneSecurityGroup: Pilih SecurityGroup nilai dari AWS CloudFormation output yang Anda hasilkan saat Anda membuat [VPC](#) Anda.

Langkah-langkah berikut menunjukkan satu operasi untuk mengambil grup yang berlaku.

- a. Buka [konsol Amazon EKS](#).
- b. Pilih nama cluster.
- c. Pilih tab Jaringan.

- d. Gunakan nilai grup keamanan tambahan sebagai referensi saat memilih dari daftar `ClusterControlPlaneSecurityGroup` tarik-turun.
- `NodeGroupName`: Masukkan nama untuk grup node Anda. Nama ini dapat digunakan nanti untuk mengidentifikasi grup node Auto Scaling yang dibuat untuk node Anda. Nama grup node tidak boleh lebih dari 63 karakter. Itu harus dimulai dengan huruf atau digit, tetapi juga dapat menyertakan tanda hubung dan garis bawah untuk karakter yang tersisa.
 - `NodeAutoScalingGroupMinSize`: Masukkan jumlah minimum node yang dapat diskalakan oleh grup Auto Scaling node Anda.
 - `NodeAutoScalingGroupDesiredCapacity`: Masukkan jumlah node yang diinginkan untuk diskalakan saat tumpukan Anda dibuat.
 - `NodeAutoScalingGroupMaxSize`: Masukkan jumlah maksimum node yang dapat diskalakan oleh grup Auto Scaling node Anda.
 - `NodeInstanceType`: Pilih jenis instance untuk node Anda. Untuk informasi selengkapnya, lihat [the section called “Jenis EC2 instans Amazon”](#).
 - `NodeImageIdSSMParam`: Diisi sebelumnya dengan parameter Amazon EC2 Systems Manager dari Amazon EKS baru-baru ini yang dioptimalkan AMI untuk versi Kubernetes variabel. [Untuk menggunakan versi minor Kubernetes berbeda yang didukung dengan Amazon EKS, ganti 1.XX dengan versi lain yang didukung](#). Sebaiknya tentukan versi Kubernetes yang sama dengan klaster Anda.

Anda juga dapat mengganti `amazon-linux-2` dengan tipe AMI yang berbeda. Untuk informasi selengkapnya, lihat [the section called “Dapatkan yang terbaru IDs”](#).

Note

Node Amazon EKS AMIs didasarkan pada Amazon Linux. Anda dapat melacak peristiwa keamanan atau privasi untuk Amazon Linux 2 di [Pusat Keamanan Amazon Linux](#) atau berlangganan ke [Umpan RSS](#) yang terkait. Kejadian keamanan dan privasi mencakup gambaran umum mengenai masalah, paket apa yang terpengaruh, dan cara memperbarui instans Anda untuk memperbaiki masalah tersebut.

- `NodeImageId`: (Opsional) Jika Anda menggunakan AMI kustom Anda sendiri (bukan AMI yang dioptimalkan Amazon EKS), masukkan ID AMI node untuk AWS Wilayah Anda. Jika Anda menentukan nilai di sini, itu akan mengganti nilai apa pun di bidang `NodeImageIdSSMParam`.
- `NodeVolumeSize`: Tentukan ukuran volume root untuk node Anda, di GiB.

- **NodeVolumeType:** Tentukan jenis volume root untuk node Anda.
- **KeyName:** Masukkan nama key pair Amazon EC2 SSH yang dapat Anda gunakan untuk terhubung menggunakan SSH ke node Anda setelah diluncurkan. Jika Anda belum memiliki EC2 key pair Amazon, Anda dapat membuatnya di AWS Management Console. Untuk informasi selengkapnya, lihat [pasangan EC2 kunci Amazon](#) di Panduan EC2 Pengguna Amazon.

 Note

Jika Anda tidak menyediakan key pair di sini, pembuatan AWS CloudFormation stack gagal.

- **BootstrapArguments:** Tentukan argumen opsional apa pun untuk diteruskan ke skrip bootstrap node, seperti kubelet argumen tambahan. Untuk informasi lebih lanjut, lihat [informasi penggunaan skrip bootstrap](#) di GitHub.

Untuk menyebarkan grup simpul yang:

- dapat menetapkan jumlah alamat IP yang jauh lebih tinggi ke Pod daripada konfigurasi default, lihat [the section called “Meningkatkan alamat IP”](#).
- dapat menetapkan IPv4 alamat ke Pod dari blok CIDR yang berbeda dari instance, lihat [the section called “Jaringan kustom”](#)
- dapat menetapkan IPv6 alamat ke Pod dan layanan, lihat [the section called “IPv6”](#).
- tidak memiliki akses internet keluar, lihat [the section called “klaster privat”](#).
- **Nonaktifkan IMDSv1:** Secara default, setiap node mendukung Layanan Metadata Instans Versi 1 (IMDSv1) dan. IMDSv2 Anda dapat menonaktifkan IMDSv1. Untuk mencegah penggunaan node dan Pod future dalam grup node MDSv1, setel Disable IMDSv1 ke true. Untuk informasi selengkapnya tentang IMDS, lihat [Mengonfigurasi layanan metadata instans](#). Untuk informasi selengkapnya tentang membatasi akses ke node Anda, lihat [Membatasi akses ke profil instance yang ditetapkan ke node pekerja](#).
- **VpcId:** Masukkan ID untuk [VPC](#) yang Anda buat.
- **Subnet:** Pilih subnet yang sudah Anda buat untuk VPC Anda. Jika Anda membuat VPC menggunakan langkah-langkah yang dijelaskan dalam [Membuat VPC Amazon untuk kluster Amazon EKS Anda](#), tentukan hanya subnet pribadi dalam VPC agar node Anda dapat diluncurkan. Anda dapat melihat subnet mana yang bersifat pribadi dengan membuka setiap subnet link dari tab Networking cluster Anda.

⚠ Important

- Jika salah satu subnet adalah subnet publik, maka mereka harus mengaktifkan pengaturan tugas alamat IP publik otomatis. Jika pengaturan tidak diaktifkan untuk subnet publik, maka node apa pun yang Anda terapkan ke subnet publik tersebut tidak akan diberi alamat IP publik dan tidak akan dapat berkomunikasi dengan cluster atau layanan lainnya. AWS Jika subnet digunakan sebelum 26 Maret 2020 menggunakan salah satu [templat AWS CloudFormation VPC Amazon EKS](#), atau dengan menggunakan `eksctl`, maka penetapan alamat IP publik otomatis dinonaktifkan untuk subnet publik. Untuk informasi tentang cara mengaktifkan penetapan alamat IP publik untuk subnet, lihat [Memodifikasi atribut IPv4 pengalamatan publik](#) untuk subnet Anda. Jika node dikerahkan ke subnet pribadi, maka node dapat berkomunikasi dengan cluster dan AWS layanan lainnya melalui gateway NAT.
- Jika subnet tidak memiliki akses internet, pastikan Anda mengetahui pertimbangan dan langkah tambahan dalam [Menyebarkan kluster pribadi dengan](#) akses internet terbatas.
- Jika Anda memilih subnet AWS Outposts, Wavelength, atau Local Zone, subnet tidak boleh diteruskan saat Anda membuat cluster.

9. Pilih pilihan yang Anda inginkan di halaman Configure stack options, lalu pilih Next.

10. Pilih kotak centang di sebelah kiri Saya mengakui yang AWS CloudFormation mungkin membuat sumber daya IAM. , dan kemudian pilih Buat tumpukan.

11. Setelah tumpukan Anda selesai dibuat, pilih tumpukan di konsol dan pilih Outputs.

12. Rekam `NodeInstanceRole` untuk grup node yang telah dibuat. Anda memerlukan ini saat mengonfigurasi simpul Amazon EKS Anda.

Langkah 2: Aktifkan node untuk bergabung dengan cluster Anda

📘 Note

Jika Anda meluncurkan node di dalam VPC pribadi tanpa akses internet keluar, pastikan untuk mengaktifkan node untuk bergabung dengan cluster Anda dari dalam VPC.

1. Periksa untuk melihat apakah Anda sudah memiliki `aws-authConfigMap`.

```
kubectl describe configmap -n kube-system aws-auth
```

2. Jika Anda ditampilkan `aws-authConfigMap`, maka perbarui sesuai kebutuhan.

a. Buka `ConfigMap` untuk mengedit.

```
kubectl edit -n kube-system configmap/aws-auth
```

b. Tambahkan `mapRoles` entri baru sesuai kebutuhan. Tetapkan `roleARN` nilai ke `NodeInstanceRole` nilai yang Anda rekam dalam prosedur sebelumnya.

```
[...]
data:
  mapRoles: |
    - roleARN: <ARN of instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
[...]
```

c. Simpan file dan keluar dari editor teks Anda.

3. Jika Anda menerima kesalahan yang menyatakan "Error from server (NotFound): configmaps "aws-auth" not found, maka terapkan `stokConfigMap`.

a. Unduh peta konfigurasi.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/
aws-auth-cm.yaml
```

b. Dalam `aws-auth-cm.yaml` file, atur `roleARN` nilai ke `NodeInstanceRole` nilai yang Anda rekam dalam prosedur sebelumnya. Anda dapat melakukan ini dengan editor teks, atau dengan mengganti `my-node-instance-role` dan menjalankan perintah berikut:

```
sed -i.bak -e 's|<ARN of instance role (not instance profile)>|my-node-instance-
role|' aws-auth-cm.yaml
```

c. Terapkan konfigurasi. Perintah ini mungkin memerlukan waktu beberapa menit untuk diselesaikan.

```
kubectl apply -f aws-auth-cm.yaml
```

4. Perhatikan status simpul Anda dan tunggu sampai simpul mencapai Status Ready.

```
kubectl get nodes --watch
```

Masukkan `Ctrl+C` untuk kembali ke prompt shell.

Note

Jika Anda menerima kesalahan otorisasi atau jenis sumber daya, lihat [the section called “Tidak sah atau akses ditolak \(kubectl\)”](#) di topik pemecahan masalah.

Jika node gagal bergabung dengan cluster, maka lihat [the section called “Simpul gagal untuk bergabung dengan kluster”](#) di chapter Troubleshooting.

5. (Hanya node GPU) Jika Anda memilih jenis instans GPU dan AMI akselerasi Amazon EKS yang dioptimalkan, Anda harus menerapkan [plugin perangkat NVIDIA untuk Kubernetes](#) sebagai a di cluster Anda. DaemonSet Ganti `vX.X.X` dengan s-device-plugin versi [NVIDIA/K8](#) yang Anda inginkan sebelum menjalankan perintah berikut.

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/vX.X.X/
deployments/static/nvidia-device-plugin.yml
```

Langkah 3: Tindakan tambahan

1. (Opsional) Deploy [aplikasi sampel](#) untuk menguji simpul kluster dan Linux Anda.
2. (Opsional) Jika kebijakan IAM terkelola `AmazonEKS_CNI_Policy` (jika Anda memiliki kluster) atau `AmazonEKS_CNI_IPv6_Policy` (yang Anda [buat sendiri](#) jika Anda memiliki IPv4 kluster) dilampirkan ke peran IAM [node Amazon EKS Anda](#), kami [sarankan untuk menentukannya ke peran IAM](#) yang Anda kaitkan ke akun layanan Kubernetes sebagai gantinya. IPv6 `aws-node` Untuk informasi selengkapnya, lihat [the section called “Konfigurasi untuk IRSA”](#).
3. Sebaiknya blokir akses Pod ke IMDS jika kondisi berikut benar:
 - Anda berencana untuk menetapkan peran IAM ke semua akun layanan Kubernetes Anda sehingga Pod hanya memiliki izin minimum yang mereka butuhkan.
 - Tidak ada Pod dalam kluster yang memerlukan akses ke layanan metadata EC2 instans Amazon (IMDS) karena alasan lain, seperti mengambil Region saat ini. AWS

Untuk informasi selengkapnya, lihat [Membatasi akses ke profil instance yang ditetapkan ke node pekerja](#).

Buat node Bottlerocket yang dikelola sendiri

Note

Grup node terkelola mungkin menawarkan beberapa keuntungan untuk kasus penggunaan Anda. Untuk informasi selengkapnya, lihat [the section called “Grup simpul terkelola”](#).

Topik ini menjelaskan cara meluncurkan grup Auto Scaling dari node [Bottlerocket](#) yang mendaftar dengan cluster Amazon EKS Anda. Bottlerocket adalah sistem operasi open-source berbasis Linux AWS yang dapat Anda gunakan untuk menjalankan kontainer pada mesin virtual atau host bare metal. Setelah simpul bergabung dengan klaster, Anda dapat men-deploy aplikasi Kubernetes pada mereka. [Untuk informasi selengkapnya tentang Bottlerocket, lihat Menggunakan AMI Bottlerocket dengan Amazon EKS aktif GitHub dan dukungan AMI Kustom dalam dokumentasi.](#) eksctl

Untuk informasi tentang peningkatan di tempat, lihat Operator Pembaruan [Bottlerocket](#) di GitHub

Important

- Node Amazon EKS adalah EC2 instans Amazon standar, dan Anda ditagih berdasarkan harga EC2 instans Amazon normal. Untuk informasi selengkapnya, lihat [EC2 harga Amazon](#).
- Anda dapat meluncurkan node Bottlerocket di Amazon EKS cluster yang diperluas di AWS Outposts, tetapi Anda tidak dapat meluncurkannya di cluster lokal di Outposts. AWS Untuk informasi selengkapnya, lihat [Amazon EKS di AWS Outposts](#).
- Anda dapat menerapkan ke EC2 instans Amazon dengan x86 atau prosesor Arm. Namun, Anda tidak dapat menyebarkan ke instance yang memiliki chip Inferentia.
- Bottlerocket kompatibel dengan AWS CloudFormation Namun, tidak ada CloudFormation template resmi yang dapat disalin untuk menyebarkan node Bottlerocket untuk Amazon EKS.
- Gambar bottlerocket tidak datang dengan server SSH atau shell. Anda dapat menggunakan metode out-of-band akses untuk memungkinkan SSH mengaktifkan wadah

admin dan meneruskan beberapa langkah konfigurasi bootstrap dengan data pengguna. Untuk informasi lebih lanjut, lihat bagian ini di [bottlerocket](#) README.md di: GitHub

- [Eksplorasi](#)
- [Kontainer admin](#)
- [Pengaturan Kubernetes](#)

Prosedur ini membutuhkan eksctl versi 0.210.0 atau yang lebih baru. Anda dapat memeriksa versi Anda dengan perintah berikut:

```
eksctl version
```

Untuk petunjuk tentang cara menginstal atau meningkatkan eksctl, lihat [Instalasi](#) dalam eksctl dokumentasi. Catatan: Prosedur ini hanya berfungsi untuk cluster yang dibuat dengan eksctl

1. Salin konten berikut ke perangkat Anda. Ganti *my-cluster* dengan nama kluster Anda. Nama hanya dapat berisi karakter alfanumerik (peka huruf besar/kecil) dan tanda hubung. Itu harus dimulai dengan karakter alfanumerik dan tidak boleh lebih dari 100 karakter. Nama harus unik di dalam AWS Wilayah dan AWS akun tempat Anda membuat kluster. Ganti *ng-bottlerocket* dengan nama untuk grup node Anda. Nama grup node tidak boleh lebih dari 63 karakter. Itu harus dimulai dengan huruf atau digit, tetapi juga dapat menyertakan tanda hubung dan garis bawah untuk karakter yang tersisa. Untuk menerapkan pada instance Arm, ganti *m5.large* dengan tipe instance Arm. Ganti *my-ec2-keypair-name* dengan nama key pair Amazon EC2 SSH yang dapat Anda gunakan untuk terhubung menggunakan SSH ke node Anda setelah diluncurkan. Jika Anda belum memiliki EC2 key pair Amazon, Anda dapat membuatnya di AWS Management Console. Untuk informasi selengkapnya, lihat [pasangan EC2 kunci Amazon](#) di Panduan EC2 Pengguna Amazon. Ganti semua yang tersisa *example values* dengan nilai Anda sendiri. Setelah Anda membuat penggantian, jalankan perintah yang dimodifikasi untuk membuat `bottlerocket.yaml` file.

Jika menentukan jenis EC2 instans Amazon Arm, tinjau pertimbangan di Amazon [EKS yang dioptimalkan Arm Amazon Linux AMIs sebelum menerapkan](#). Untuk petunjuk tentang cara menerapkan menggunakan AMI kustom, lihat [Membangun Bottlerocket](#) dan dukungan [AMI GitHub](#) Kustom dalam dokumentasi. eksctl Untuk menerapkan grup node terkelola, terapkan AMI kustom menggunakan template peluncuran. Untuk informasi selengkapnya, lihat [the section called “Templat peluncuran”](#).

 Important

Untuk menyebarkan grup node ke subnet AWS Outposts, AWS Wavelength, atau AWS Local Zone, jangan lewatkan subnet AWS Outposts, Wavelength, atau Local Zone saat Anda membuat AWS cluster. AWS Anda harus menentukan subnet dalam contoh berikut. Untuk informasi selengkapnya lihat [Buat nodegroup dari file config](#) dan [Skema file config](#) di dalam dokumentasi eksctl. Ganti *region-code* dengan AWS Wilayah tempat cluster Anda berada.

```
cat >bottlerocket.yaml <<EOF
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code
  version: '1.33'

iam:
  withOIDC: true

nodeGroups:
- name: ng-bottlerocket
  instanceType: m5.large
  desiredCapacity: 3
  amiFamily: Bottlerocket
  ami: auto-ssm
  iam:
    attachPolicyARNs:
      - arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy
      - arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
      - arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
      - arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
  ssh:
    allow: true
    publicKeyName: my-ec2-keypair-name
EOF
```

2. Deploy simpul Anda dengan perintah berikut.

```
eksctl create nodegroup --config-file=bottlerocket.yaml
```

Contoh output adalah sebagai berikut.

Beberapa baris adalah output sementara node dibuat. Salah satu baris terakhir output adalah baris contoh berikut.

```
[#] created 1 nodegroup(s) in cluster "my-cluster"
```

3. (Opsional) Buat [volume persisten](#) Kubernetes pada simpul Bottlerocket menggunakan [Plugin Amazon EBS CSI](#). Driver Amazon EBS default bergantung pada alat sistem file yang tidak disertakan dengan Bottlerocket. Untuk informasi selengkapnya tentang cara membuat kelas penyimpanan menggunakan driver, lihat [the section called "Amazon EBS"](#).
4. (Opsional) Secara default, kube-proxy menetapkan parameter `nf_conntrack_max` kernel ke nilai default yang mungkin berbeda dari apa yang awalnya ditetapkan Bottlerocket saat boot. Untuk menjaga [pengaturan default](#) Bottlerocket, edit kube-proxy konfigurasi dengan perintah berikut.

```
kubectyl edit -n kube-system daemonset kube-proxy
```

Tambahkan `--conntrack-max-per-core` dan `--conntrack-min` ke kube-proxy argumen yang ada dalam contoh berikut. Pengaturan `0` menyiratkan bahwa tidak ada perubahan.

```
containers:
- command:
  - kube-proxy
  - --v=2
  - --config=/var/lib/kube-proxy-config/config
  - --conntrack-max-per-core=0
  - --conntrack-min=0
```

5. (Opsional) Deploy [aplikasi sampel](#) untuk menguji simpul Bottlerocket Anda.
6. Sebaiknya blokir akses Pod ke IMDS jika kondisi berikut benar:
 - Anda berencana untuk menetapkan peran IAM ke semua akun layanan Kubernetes Anda sehingga Pod hanya memiliki izin minimum yang mereka butuhkan.
 - Tidak ada Pod dalam kluster yang memerlukan akses ke layanan metadata EC2 instans Amazon (IMDS) karena alasan lain, seperti mengambil Region saat ini. AWS

Untuk informasi selengkapnya, lihat [Membatasi akses ke profil instance yang ditetapkan ke node pekerja](#).

Buat node Microsoft Windows yang dikelola sendiri

Topik ini menjelaskan cara meluncurkan grup Auto Scaling dari node Windows yang mendaftar dengan kluster Amazon EKS Anda. Setelah simpul bergabung dengan kluster, Anda dapat men-deploy aplikasi Kubernetes pada mereka.

Important

- Node Amazon EKS adalah EC2 instans Amazon standar, dan Anda ditagih berdasarkan harga EC2 instans Amazon normal. Untuk informasi selengkapnya, lihat [EC2 harga Amazon](#).
- Anda dapat meluncurkan node Windows di Amazon EKS cluster yang diperluas di AWS Outposts, tetapi Anda tidak dapat meluncurkannya di cluster AWS lokal di Outposts. Untuk informasi selengkapnya, lihat [Amazon EKS di AWS Outposts](#).

Aktifkan dukungan Windows untuk cluster Anda. Kami menyarankan Anda meninjau pertimbangan penting sebelum meluncurkan grup node Windows. Untuk informasi selengkapnya, lihat [the section called “Aktifkan dukungan Windows”](#).

Anda dapat meluncurkan node Windows yang dikelola sendiri dengan salah satu dari berikut ini:

- [the section called “eksctl”](#)
- [the section called “AWS Management Console”](#)

eksctl

Luncurkan node Windows yang dikelola sendiri menggunakan **eksctl**

Prosedur ini mengharuskan Anda untuk sudah menginstal eksctl, dan bahwa eksctl versi yang Anda miliki setidaknya 0.210.0. Anda dapat memeriksa versi Amazon Linux Anda dengan perintah berikut.

```
eksctl version
```

Untuk petunjuk tentang cara menginstal atau meningkatkan `eksctl`, lihat [Instalasi](#) dalam `eksctl` dokumentasi.

 Note

Prosedur ini hanya bekerja untuk klaster yang dibuat dengan `eksctl`.

1. (Opsional) Jika kebijakan IAM terkelola `AmazonEKS_CNI_Policy` (jika Anda memiliki klaster) atau `AmazonEKS_CNI_IPv6_Policy` (yang Anda [buat sendiri](#) jika Anda memiliki IPv4 klaster) dilampirkan ke peran IAM [node Amazon EKS Anda](#), kami sarankan untuk [menetapkannya ke peran IAM](#) yang Anda kaitkan ke akun layanan Kubernetes sebagai gantinya. IPv6 `aws-node` Untuk informasi selengkapnya, lihat [the section called “Konfigurasi untuk IRSA”](#).
2. Prosedur ini mengasumsikan bahwa Anda memiliki cluster yang ada. Jika Anda belum memiliki kluster Amazon EKS dan grup node Amazon Linux untuk menambahkan grup node Windows, kami sarankan Anda mengikuti [the section called “Buat cluster \(eksctl\)”](#). Panduan ini memberikan panduan lengkap tentang cara membuat cluster Amazon EKS dengan node Amazon Linux.

Buat grup simpul Anda dengan perintah berikut. Ganti `region-code` dengan AWS Wilayah tempat cluster Anda berada. Ganti `my-cluster` dengan nama klaster Anda. Nama hanya dapat berisi karakter alfanumerik (peka huruf besar/kecil) dan tanda hubung. Itu harus dimulai dengan karakter alfanumerik dan tidak boleh lebih dari 100 karakter. Nama harus unik di dalam AWS Wilayah dan AWS akun tempat Anda membuat klaster. Ganti `ng-windows` dengan nama untuk grup node Anda. Nama grup node tidak boleh lebih dari 63 karakter. Itu harus dimulai dengan huruf atau digit, tetapi juga dapat menyertakan tanda hubung dan garis bawah untuk karakter yang tersisa. Anda dapat mengganti `2019` dengan `2022` menggunakan Windows Server 2022. Ganti sisanya `example values` dengan nilai-nilai Anda sendiri.

 Important

Untuk menyebarkan grup node ke subnet AWS Outposts, AWS Wavelength, atau Local Zone, jangan lewatkan subnet AWS Outposts, Wavelength, AWS atau Local Zone saat Anda membuat cluster. Buat grup node dengan file konfigurasi, tentukan subnet AWS Outposts, Wavelength, atau Local Zone. Untuk informasi selengkapnya, lihat [Membuat nodegroup dari file konfigurasi dan skema file Config](#) dalam dokumentasi. `eksctl`

```
eksctl create nodegroup \  
  --region region-code \  
  --cluster my-cluster \  
  --name ng-windows \  
  --node-type t2.large \  
  --nodes 3 \  
  --nodes-min 1 \  
  --nodes-max 4 \  
  --managed=false \  
  --node-ami-family WindowsServer2019FullContainer
```

Note

- Jika simpul gagal bergabung dengan kluster, lihat [the section called “Simpul gagal untuk bergabung dengan kluster”](#) dalam panduan Pemecahan Masalah.
- Untuk melihat opsi yang tersedia untuk eksctl perintah, masukkan perintah berikut.

```
eksctl command -help
```

Contoh output adalah sebagai berikut. Beberapa baris adalah output sementara node dibuat. Salah satu baris terakhir dari output adalah baris contoh berikutnya.

```
[#] created 1 nodegroup(s) in cluster "my-cluster"
```

3. (Opsional) Deploy [aplikasi sampel](#) untuk menguji simpul kluster dan Windows Anda.
4. Sebaiknya blokir akses Pod ke IMDS jika kondisi berikut benar:
 - Anda berencana untuk menetapkan peran IAM ke semua akun layanan Kubernetes Anda sehingga Pod hanya memiliki izin minimum yang mereka butuhkan.
 - Tidak ada Pod dalam kluster yang memerlukan akses ke layanan metadata EC2 instans Amazon (IMDS) karena alasan lain, seperti mengambil Region saat ini. AWS

Untuk informasi selengkapnya, lihat [Membatasi akses ke profil instance yang ditetapkan ke node pekerja](#).

AWS Management Console

Prasyarat

- Kluster Amazon EKS yang sudah ada dan grup simpul Linux. Jika Anda tidak memiliki sumber daya ini, kami sarankan Anda membuatnya menggunakan salah satu panduan kami di [Memulai](#). Panduan ini menjelaskan cara membuat cluster Amazon EKS dengan node Linux.
- Grup VPC dan keamanan yang ada yang memenuhi persyaratan untuk kluster Amazon EKS. Untuk informasi selengkapnya, lihat [the section called “Persyaratan VPC dan subnet”](#) dan [the section called “Persyaratan grup keamanan”](#). Panduan dalam [Memulai](#) membuat VPC yang memenuhi persyaratan. Atau, Anda juga dapat mengikuti [Buat VPC Amazon untuk kluster Amazon EKS](#) Anda untuk membuatnya secara manual.
- Cluster Amazon EKS yang ada yang menggunakan VPC dan grup keamanan yang memenuhi persyaratan kluster Amazon EKS. Untuk informasi selengkapnya, lihat [the section called “Membuat kluster”](#). Jika Anda memiliki subnet di AWS Wilayah di mana Anda mengaktifkan AWS Outposts, AWS Wavelength, atau AWS Local Zones, subnet tersebut tidak boleh diteruskan saat Anda membuat cluster.

Langkah 1: Luncurkan node Windows yang dikelola sendiri menggunakan AWS Management Console

1. Tunggu status kluster Anda ditampilkan sebagai ACTIVE. Jika Anda meluncurkan node sebelum cluster aktif, node gagal mendaftar dengan cluster dan Anda perlu meluncurkannya kembali.
2. Buka [konsol AWS CloudFormation](#)
3. Pilih Buat tumpukan.
4. Untuk Tentukan templat, pilih Amazon S3 URL.
5. Salin URL berikut dan tempel ke URL Amazon S3.

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2023-02-09/amazon-eks-windows-nodegroup.yaml
```

6. Pilih Berikutnya dua kali.
7. Pada halaman Quick create stack, masukkan parameter berikut yang sesuai:
 - Nama tumpukan: Pilih nama tumpukan untuk AWS CloudFormation tumpukan Anda. Misalnya, Anda bisa menyebutny `my-cluster-nodes`.
 - ClusterName: Masukkan nama yang Anda gunakan saat membuat cluster Amazon EKS Anda.

 Important

Nama ini harus sama persis dengan nama yang Anda gunakan di [Langkah 1: Buat cluster Amazon EKS Anda](#). Jika tidak, node Anda tidak dapat bergabung dengan cluster.

- **ClusterControlPlaneSecurityGroup:** Pilih grup keamanan dari AWS CloudFormation output yang Anda hasilkan saat membuat [VPC](#) Anda. Langkah-langkah berikut menunjukkan satu metode untuk mengambil grup yang berlaku.
 - a. Buka [konsol Amazon EKS](#).
 - b. Pilih nama cluster.
 - c. Pilih tab Jaringan.
 - d. Gunakan nilai grup keamanan tambahan sebagai referensi saat memilih dari daftar ClusterControlPlaneSecurityGroup tarik-turun.
- **NodeGroupName:** Masukkan nama untuk grup node Anda. Nama ini dapat digunakan nanti untuk mengidentifikasi grup node Auto Scaling yang dibuat untuk node Anda. Nama grup node tidak boleh lebih dari 63 karakter. Itu harus dimulai dengan huruf atau digit, tetapi juga dapat menyertakan tanda hubung dan garis bawah untuk karakter yang tersisa.
- **NodeAutoScalingGroupMinSize:** Masukkan jumlah minimum node yang dapat diskalakan oleh grup Auto Scaling node Anda.
- **NodeAutoScalingGroupDesiredCapacity:** Masukkan jumlah node yang diinginkan untuk diskalakan saat tumpukan Anda dibuat.
- **NodeAutoScalingGroupMaxSize:** Masukkan jumlah maksimum node yang dapat diskalakan oleh grup Auto Scaling node Anda.
- **NodeInstanceType:** Pilih jenis instance untuk node Anda. Untuk informasi selengkapnya, lihat [the section called “Jenis EC2 instans Amazon”](#).

 Note

[Tipe instans yang didukung untuk versi terbaru plugin Amazon VPC CNI untuk Kubernetes tercantum di `vpc\_ip\_resource\_limit.go on`](#). GitHub Anda mungkin perlu memperbarui versi CNI untuk menggunakan jenis instans terbaru yang didukung. Untuk informasi selengkapnya, lihat [the section called “Amazon VPC CNI”](#).

- `NodeImageIdSSMParam`: Diisi sebelumnya dengan parameter Amazon EC2 Systems Manager dari ID AMI Windows Core Windows Core Amazon EKS yang direkomendasikan saat ini. Untuk menggunakan versi lengkap Windows, ganti `Core` dengan `Full`.
- `NodeImageId`: (Opsional) Jika Anda menggunakan AMI kustom Anda sendiri (bukan AMI yang dioptimalkan Amazon EKS), masukkan ID AMI node untuk AWS Wilayah Anda. Jika Anda menentukan nilai untuk bidang ini, itu akan menggantikan nilai apa pun di `NodeImageIdSSMParam` bidang.
- `NodeVolumeSize`: Tentukan ukuran volume root untuk node Anda, di GiB.
- `KeyName`: Masukkan nama key pair Amazon EC2 SSH yang dapat Anda gunakan untuk terhubung menggunakan SSH ke node Anda setelah diluncurkan. Jika Anda belum memiliki EC2 key pair Amazon, Anda dapat membuatnya di AWS Management Console. Untuk informasi selengkapnya, lihat [pasangan EC2 kunci Amazon](#) di Panduan EC2 Pengguna Amazon.

 Note

Jika Anda tidak menyediakan key pair di sini, AWS CloudFormation tumpukan gagal dibuat.

- `BootstrapArguments`: Tentukan argumen opsional apa pun untuk diteruskan ke skrip bootstrap node, seperti `kubelet` argumen tambahan yang digunakan `KubeletExtraArgs`.
- Nonaktifkan IMDSv1: Secara default, setiap node mendukung Layanan Metadata Instans Versi 1 (IMDSv1) dan IMDSv2. Anda dapat menonaktifkan IMDSv1. Untuk mencegah penggunaan node dan Pod future dalam grup node IMDSv1, setel `DisableIMDSv1` ke `true`. Untuk informasi selengkapnya tentang IMDS, lihat [Mengonfigurasi layanan metadata instans](#).
- `VpcId`: Pilih ID untuk [VPC](#) yang Anda buat.
- `NodeSecurityGroups`: Pilih grup keamanan yang dibuat untuk grup node Linux Anda saat Anda membuat [VPC](#) Anda. Jika node Linux Anda memiliki lebih dari satu grup keamanan yang melekat padanya, tentukan semuanya. Ini untuk, misalnya, jika grup node Linux dibuat dengan `eksctl`.
- `Subnet`: Pilih subnet yang Anda buat. Jika Anda membuat VPC menggunakan langkah-langkah dalam [Membuat VPC Amazon untuk kluster Amazon EKS Anda](#), tentukan hanya subnet pribadi dalam VPC untuk node Anda untuk diluncurkan.

⚠ Important

- Jika salah satu subnet adalah subnet publik, maka mereka harus mengaktifkan pengaturan tugas alamat IP publik otomatis. Jika pengaturan tidak diaktifkan untuk subnet publik, maka node apa pun yang Anda terapkan ke subnet publik tersebut tidak akan diberi alamat IP publik dan tidak akan dapat berkomunikasi dengan cluster atau layanan lainnya. AWS Jika subnet digunakan sebelum 26 Maret 2020 menggunakan salah satu [templat AWS CloudFormation VPC Amazon EKS](#), atau dengan menggunakan `eksctl`, maka penetapan alamat IP publik otomatis dinonaktifkan untuk subnet publik. Untuk informasi tentang cara mengaktifkan penetapan alamat IP publik untuk subnet, lihat [Memodifikasi atribut IPv4 pengalamatan publik](#) untuk subnet Anda. Jika node diarahkan ke subnet pribadi, maka node dapat berkomunikasi dengan cluster dan AWS layanan lainnya melalui gateway NAT.
- Jika subnet tidak memiliki akses internet, pastikan Anda mengetahui pertimbangan dan langkah tambahan dalam [Menyebarkan kluster pribadi dengan](#) akses internet terbatas.
- Jika Anda memilih subnet AWS Outposts, Wavelength, atau Local Zone, maka subnet tidak boleh diteruskan saat Anda membuat cluster.

8. Nyatakan bahwa tumpukan dapat membuat sumber daya IAM, kemudian pilih Buat tumpukan.
9. Setelah tumpukan Anda selesai dibuat, pilih tumpukan di konsol dan pilih Outputs.
10. Rekam `NodeInstanceRole` untuk grup node yang telah dibuat. Anda memerlukan ini saat mengonfigurasi simpul Amazon EKS Windows Anda.

Langkah 2: Aktifkan node untuk bergabung dengan cluster Anda

1. Periksa untuk melihat apakah Anda sudah memiliki `aws-authConfigMap`.

```
kubectl describe configmap -n kube-system aws-auth
```

2. Jika Anda ditampilkan `aws-authConfigMap`, maka perbarui sesuai kebutuhan.
 - a. Buka `ConfigMap` untuk mengedit.

```
kubectl edit -n kube-system configmap/aws-auth
```

- b. Tambahkan mapRoles entri baru sesuai kebutuhan. Tetapkan rolearn nilai ke NodeInstanceRole yang Anda rekam dalam prosedur sebelumnya.

```
[...]
data:
  mapRoles: |
    - rolearn: <ARN of linux instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
    - rolearn: <ARN of windows instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
        - eks:kube-proxy-windows
[...]
```

- c. Simpan file dan keluar dari editor teks Anda.

3. Jika Anda menerima kesalahan yang menyatakan "Error from server (NotFound): configmaps "aws-auth" not found, maka terapkan stokConfigMap.

- a. Unduh peta konfigurasi.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/
aws-auth-cm-windows.yaml
```

- b. Dalam aws-auth-cm-windows.yaml file, atur rolearn nilai ke nilai yang berlaku NodeInstanceRole yang Anda rekam dalam prosedur sebelumnya. Anda dapat melakukan ini dengan editor teks, atau dengan mengganti *example values* dan menjalankan perintah berikut:

```
sed -i.bak -e 's|<ARN of linux instance role (not instance profile)>|my-node-
linux-instance-role|' \
  -e 's|<ARN of windows instance role (not instance profile)>|my-node-windows-
instance-role|' aws-auth-cm-windows.yaml
```

Important

- Jangan mengubah baris lain dalam file ini.

- Jangan gunakan peran IAM yang sama untuk node Windows dan Linux.

- c. Terapkan konfigurasi. Perintah ini mungkin memakan waktu beberapa menit untuk menyelesaikannya.

```
kubectl apply -f aws-auth-cm-windows.yaml
```

4. Perhatikan status simpul Anda dan tunggu sampai simpul mencapai Status Ready.

```
kubectl get nodes --watch
```

Masukkan `Ctrl+C` untuk kembali ke prompt shell.

Note

Jika Anda menerima kesalahan otorisasi atau jenis sumber daya, lihat [the section called “Tidak sah atau akses ditolak \(kubectl\)”](#) di topik pemecahan masalah.

Jika node gagal bergabung dengan cluster, maka lihat [the section called “Simpul gagal untuk bergabung dengan kluster”](#) di chapter Troubleshooting.

Langkah 3: Tindakan tambahan

1. (Opsional) Deploy [aplikasi sampel](#) untuk menguji simpul kluster dan Windows Anda.
2. (Opsional) Jika kebijakan IAM terkelola `AmazonEKS_CNI_Policy` (jika Anda memiliki kluster) atau `AmazonEKS_CNI_IPv6_Policy` (yang Anda [buat sendiri](#) jika Anda memiliki IPv4 kluster) dilampirkan ke peran IAM [node Amazon EKS Anda](#), kami sarankan untuk [menetapkannya ke peran IAM](#) yang Anda kaitkan ke akun layanan Kubernetes sebagai gantinya. IPv6 `aws-node` Untuk informasi selengkapnya, lihat [the section called “Konfigurasi untuk IRSA”](#).
3. Sebaiknya blokir akses Pod ke IMDS jika kondisi berikut benar:
 - Anda berencana untuk menetapkan peran IAM ke semua akun layanan Kubernetes Anda sehingga Pod hanya memiliki izin minimum yang mereka butuhkan.
 - Tidak ada Pod dalam kluster yang memerlukan akses ke layanan metadata EC2 instans Amazon (IMDS) karena alasan lain, seperti mengambil Region saat ini. AWS

Untuk informasi selengkapnya, lihat [Membatasi akses ke profil instance yang ditetapkan ke node pekerja](#).

Buat node Linux Ubuntu yang dikelola sendiri

Note

Grup node terkelola mungkin menawarkan beberapa keuntungan untuk kasus penggunaan Anda. Untuk informasi selengkapnya, lihat [the section called “Grup simpel terkelola”](#).

Topik ini menjelaskan cara meluncurkan grup Auto Scaling [Ubuntu di Amazon Elastic Kubernetes Service \(EKS\)](#) atau [Ubuntu Pro di node Amazon Elastic Kubernetes Service \(EKS\) yang mendaftar dengan cluster Amazon EKS](#) Anda. Ubuntu dan Ubuntu Pro untuk EKS didasarkan pada Ubuntu Minimal LTS resmi, termasuk AWS kernel khusus yang dikembangkan bersama AWS, dan telah dibangun khusus untuk EKS. Ubuntu Pro menambahkan cakupan keamanan tambahan dengan mendukung periode dukungan EKS yang diperpanjang, livepatch kernel, kepatuhan FIPS, dan kemampuan untuk menjalankan kontainer Pro tanpa batas.

Setelah node bergabung dengan cluster, Anda dapat menyebarkan aplikasi kontainer ke mereka. Untuk informasi lebih lanjut, kunjungi dokumentasi untuk [Ubuntu on AWS](#) dan [dukungan Custom AMI](#) dalam eksctl dokumentasi.

Important

- Node Amazon EKS adalah EC2 instans Amazon standar, dan Anda ditagih berdasarkan harga EC2 instans Amazon normal. Untuk informasi selengkapnya, lihat [EC2 harga Amazon](#).
- Anda dapat meluncurkan node Ubuntu di Amazon EKS cluster yang diperluas di AWS Outposts, tetapi Anda tidak dapat meluncurkannya di cluster AWS lokal di Outposts. Untuk informasi selengkapnya, lihat [Amazon EKS di AWS Outposts](#).
- Anda dapat menerapkan ke EC2 instans Amazon dengan x86 atau prosesor Arm. Namun, contoh yang memiliki chip Inferentia mungkin perlu menginstal [Neuron SDK](#) terlebih dahulu.

Prosedur ini membutuhkan eksctl versi 0.210.0 atau yang lebih baru. Anda dapat memeriksa versi Anda dengan perintah berikut:

```
eksctl version
```

Untuk petunjuk tentang cara menginstal atau meningkatkan eksctl, lihat [Instalasi](#) dalam eksctl dokumentasi. Catatan: Prosedur ini hanya berfungsi untuk cluster yang dibuat dengan eksctl

1. Salin konten berikut ke perangkat Anda. Ganti `my-cluster` dengan nama kluster Anda. Nama hanya dapat berisi karakter alfanumerik (peka huruf besar/kecil) dan tanda hubung. Itu harus dimulai dengan karakter alfabet dan tidak boleh lebih dari 100 karakter. Ganti `ng-ubuntu` dengan nama untuk grup node Anda. Nama grup node tidak boleh lebih dari 63 karakter. Itu harus dimulai dengan huruf atau digit, tetapi juga dapat menyertakan tanda hubung dan garis bawah untuk karakter yang tersisa. Untuk menerapkan pada instance Arm, ganti `m5.large` dengan tipe instance Arm. Ganti `my-ec2-keypair-name` dengan nama key pair Amazon EC2 SSH yang dapat Anda gunakan untuk terhubung menggunakan SSH ke node Anda setelah diluncurkan. Jika Anda belum memiliki EC2 key pair Amazon, Anda dapat membuatnya di AWS Management Console. Untuk informasi selengkapnya, lihat [pasangan EC2 kunci Amazon](#) di Panduan EC2 Pengguna Amazon. Ganti semua yang tersisa *example values* dengan nilai Anda sendiri. Setelah Anda membuat penggantian, jalankan perintah yang dimodifikasi untuk membuat `ubuntu.yaml` file.

Important

Untuk menyebarkan grup node ke subnet AWS Outposts, AWS Wavelength, atau AWS Local Zone, jangan lewatkan subnet AWS Outposts, Wavelength, atau Local Zone saat Anda membuat AWS cluster. AWS Anda harus menentukan subnet dalam contoh berikut. Untuk informasi selengkapnya lihat [Buat nodegroup dari file config](#) dan [Skema file config](#) di dalam dokumentasi eksctl. Ganti *region-code* dengan AWS Wilayah tempat cluster Anda berada.

```
cat >ubuntu.yaml <<EOF
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
```

```

name: my-cluster
region: region-code
version: '1.33'

iam:
  withOIDC: true

nodeGroups:
  - name: ng-ubuntu
    instanceType: m5.large
    desiredCapacity: 3
    amiFamily: Ubuntu2204
    iam:
      attachPolicyARNs:
        - arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy
        - arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
        - arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
        - arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
    ssh:
      allow: true
      publicKeyName: my-ec2-keypair-name
EOF

```

Untuk membuat grup node Ubuntu Pro, cukup ubah `amiFamily` nilainya menjadi `UbuntuPro2204`.

2. Deploy simpul Anda dengan perintah berikut.

```
eksctl create nodegroup --config-file=ubuntu.yaml
```

Contoh output adalah sebagai berikut.

Beberapa baris adalah output sementara node dibuat. Salah satu baris terakhir dari output adalah baris contoh berikutnya.

```
[#] created 1 nodegroup(s) in cluster "my-cluster"
```

3. (Opsional) Menyebarkan [aplikasi sampel](#) untuk menguji node Ubuntu Anda.

4. Sebaiknya blokir akses Pod ke IMDS jika kondisi berikut benar:

- Anda berencana untuk menetapkan peran IAM ke semua akun layanan Kubernetes Anda sehingga Pod hanya memiliki izin minimum yang mereka butuhkan.

- Tidak ada Pod dalam klaster yang memerlukan akses ke layanan metadata EC2 instans Amazon (IMDS) karena alasan lain, seperti mengambil Region saat ini. AWS

Untuk informasi selengkapnya, lihat [Membatasi akses ke profil instance yang ditetapkan ke node pekerja](#).

Perbarui node yang dikelola sendiri untuk klaster Anda

Saat AMI Amazon EKS baru yang dioptimalkan dirilis, pertimbangkan untuk mengganti node di grup node yang dikelola sendiri dengan AMI baru. Demikian juga, jika Anda telah memperbarui versi Kubernetes untuk cluster Amazon EKS Anda, perbarui node untuk menggunakan node dengan versi Kubernetes yang sama.

Important

Topik ini mencakup pembaruan simpul untuk simpul yang dikelola sendiri. Jika Anda menggunakan [grup node terkelola](#), lihat [the section called “Perbarui”](#).

Ada dua cara dasar untuk memperbarui grup simpul yang dikelola sendiri di klaster Anda untuk menggunakan AMI baru:

[Migrasikan aplikasi ke grup node baru](#)

Buat grup node baru dan migrasi Pod Anda ke grup tersebut. Migrasi ke grup node baru lebih anggun daripada sekadar memperbarui ID AMI di tumpukan yang ada AWS CloudFormation . Ini karena proses migrasi [menodai](#) grup node lama NoSchedule dan menguras node setelah tumpukan baru siap menerima beban kerja Pod yang ada.

[Perbarui tumpukan AWS CloudFormation simpul](#)

Perbarui AWS CloudFormation tumpukan untuk grup node yang ada untuk menggunakan AMI baru. Metode ini tidak didukung untuk grup node yang dibuat dengan eksctl.

Migrasikan aplikasi ke grup node baru

Topik ini menjelaskan bagaimana Anda dapat membuat grup node baru, dengan anggun memigrasikan aplikasi yang ada ke grup baru, dan menghapus grup node lama dari cluster Anda. Anda dapat bermigrasi ke grup simpul baru menggunakan eksctl atau AWS Management Console.

- [the section called “eksctl”](#)
- [the section called “AWS Management Console dan AWS CLI”](#)

eksctl

Migrasikan aplikasi Anda ke grup node baru dengan **eksctl**

Untuk informasi selengkapnya tentang penggunaan eksctl untuk migrasi, lihat [Nodegroup tidak dikelola](#) dalam dokumentasi. eksctl

Prosedur ini membutuhkan eksctl versi 0.210.0 atau yang lebih baru. Anda dapat memeriksa versi Anda dengan perintah berikut:

```
eksctl version
```

Untuk petunjuk tentang cara menginstal atau meningkatkan eksctl, lihat [Instalasi](#) dalam eksctl dokumentasi.

Note

Prosedur ini hanya bekerja untuk grup klaster dan simpul yang dibuat dengan eksctl.

1. Ambil nama grup node yang ada, ganti *my-cluster* dengan nama cluster Anda.

```
eksctl get nodegroups --cluster=my-cluster
```

Contoh output adalah sebagai berikut.

CLUSTER	NODEGROUP	CREATED	MIN SIZE	MAX SIZE
DESIRED CAPACITY	INSTANCE TYPE	IMAGE ID		
default	standard-nodes	2019-05-01T22:26:58Z	1	4
	t3.medium	ami-05a71d034119ffc12		3

2. Luncurkan grup node baru eksctl dengan perintah berikut. Dalam perintah, ganti setiap *example value* dengan nilai Anda sendiri. Nomor versi tidak boleh lebih lambat dari versi Kubernetes untuk pesawat kontrol Anda. Selain itu, tidak boleh lebih dari dua versi minor lebih awal dari versi Kubernetes untuk pesawat kontrol Anda. Kami menyarankan Anda menggunakan versi yang sama dengan pesawat kontrol Anda.

Sebaiknya blokir akses Pod ke IMDS jika kondisi berikut benar:

- Anda berencana untuk menetapkan peran IAM ke semua akun layanan Kubernetes Anda sehingga Pod hanya memiliki izin minimum yang mereka butuhkan.
- Tidak ada Pod dalam kluster yang memerlukan akses ke layanan metadata EC2 instans Amazon (IMDS) karena alasan lain, seperti mengambil Region saat ini. AWS

Untuk informasi selengkapnya, lihat [Membatasi akses ke profil instance yang ditetapkan ke node pekerja](#).

Untuk memblokir akses Pod ke IMDS, tambahkan `--disable-pod-imds` opsi ke perintah berikut.

 Note

Untuk lebih banyak flag yang tersedia dan deskripsinya, lihat <https://eksctl.io/>.

```
eksctl create nodegroup \  
  --cluster my-cluster \  
  --version 1.33 \  
  --name standard-nodes-new \  
  --node-type t3.medium \  
  --nodes 3 \  
  --nodes-min 1 \  
  --nodes-max 4 \  
  --managed=false
```

3. Setelah perintah sebelumnya selesai, verifikasi bahwa semua simpul Anda telah mencapai status Ready dengan perintah berikut:

```
kubectl get nodes
```

4. Hapus grup node asli dengan perintah berikut. Dalam perintah, ganti setiap *example value* dengan nama grup cluster dan node Anda:

```
eksctl delete nodegroup --cluster my-cluster --name standard-nodes-old
```

AWS Management Console dan AWS CLI

Migrasikan aplikasi Anda ke grup node baru dengan AWS Management Console dan CLI AWS

1. Luncurkan grup node baru dengan mengikuti langkah-langkah yang diuraikan dalam [Buat node Amazon Linux yang dikelola sendiri](#).
2. Setelah tumpukan Anda selesai dibuat, pilih tumpukan di konsol dan pilih Outputs.
3. Rekam `NodeInstanceRole` untuk grup node yang telah dibuat. Anda memerlukan ini untuk menambahkan simpul Amazon EKS baru ke kluster Anda.

Note

Jika Anda melampirkan kebijakan IAM tambahan ke peran IAM grup node lama Anda, lampirkan kebijakan yang sama tersebut ke peran IAM grup node baru Anda untuk mempertahankan fungsionalitas tersebut pada grup baru. Ini berlaku bagi Anda jika Anda menambahkan izin untuk [Kubernetes Cluster Autoscaler](#), misalnya.

4. Perbarui grup keamanan untuk kedua grup simpul sehingga mereka dapat berkomunikasi satu sama lain. Untuk informasi selengkapnya, lihat [the section called “Persyaratan grup keamanan”](#).
 - a. Rekam grup keamanan IDs untuk kedua grup node. Ini ditampilkan sebagai `NodeSecurityGroup` nilai dalam output AWS CloudFormation stack.

Anda dapat menggunakan perintah AWS CLI berikut untuk mendapatkan grup keamanan IDs dari nama tumpukan. Dalam perintah ini, `oldNodes` adalah nama AWS CloudFormation tumpukan untuk tumpukan node lama Anda, dan `newNodes` merupakan nama tumpukan tempat Anda bermigrasi. Ganti setiap *example value* dengan nilai-nilai Anda sendiri.

```
oldNodes="old_node_CFN_stack_name"
newNodes="new_node_CFN_stack_name"

oldSecGroup=$(aws cloudformation describe-stack-resources --stack-name $oldNodes \
--query 'StackResources[?
ResourceType=='AWS::EC2::SecurityGroup'].PhysicalResourceId' \
--output text)
newSecGroup=$(aws cloudformation describe-stack-resources --stack-name $newNodes \
--query 'StackResources[?
ResourceType=='AWS::EC2::SecurityGroup'].PhysicalResourceId' \
--output text)
```

- b. Tambahkan aturan masuk ke setiap grup keamanan simpul sehingga mereka menerima lalu lintas dari satu sama lain.

Perintah AWS CLI berikut menambahkan aturan masuk ke setiap grup keamanan yang memungkinkan semua lalu lintas pada semua protokol dari grup keamanan lainnya. Konfigurasi ini memungkinkan Pod di setiap grup node untuk berkomunikasi satu sama lain saat Anda memigrasikan beban kerja Anda ke grup baru.

```
aws ec2 authorize-security-group-ingress --group-id $oldSecGroup \
--source-group $newSecGroup --protocol -1
aws ec2 authorize-security-group-ingress --group-id $newSecGroup \
--source-group $oldSecGroup --protocol -1
```

5. Edit configmap `aws-auth` untuk memetakan peran instans simpul baru di RBAC.

```
kubectl edit configmap -n kube-system aws-auth
```

Tambahkan entri `mapRoles` baru untuk grup simpul baru.

```
apiVersion: v1
data:
  mapRoles: |
    - rolearn: ARN of instance role (not instance profile)
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes>
    - rolearn: arn:aws:iam::111122223333:role/nodes-1-16-NodeInstanceRole-
U11V27W93CX5
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
```

Ganti *ARN of instance role (not instance profile)* cuplikan dengan `NodeInstanceRole` nilai yang Anda rekam pada langkah [sebelumnya](#). Kemudian, simpan dan tutup file untuk menerapkan configmap yang diperbarui.

6. Perhatikan status simpul Anda dan tunggu simpul baru Anda bergabung dengan kluster dan mencapai status Ready tersebut.

```
kubectl get nodes --watch
```

7. (Opsional) Jika Anda menggunakan [Kubernetes Cluster Autoscaler](#), turunkan skala deployment ke nol (0) replika untuk menghindari tindakan penskalaan yang bertentangan.

```
kubectl scale deployments/cluster-autoscaler --replicas=0 -n kube-system
```

8. Gunakan perintah berikut untuk mencemari setiap node yang ingin Anda hapus. NoSchedule Hal ini agar Pod baru tidak dijadwalkan atau dijadwal ulang pada node yang Anda ganti. Untuk informasi selengkapnya, lihat [Taints and Tolerations dalam dokumentasi](#) Kubernetes.

```
kubectl taint nodes node_name key=value:NoSchedule
```

Jika Anda memutakhirkan node Anda ke versi Kubernetes baru, Anda dapat mengidentifikasi dan mencemari semua node dari versi Kubernetes tertentu (dalam hal ini,) dengan cuplikan kode berikut. 1.31 Nomor versi tidak boleh lebih lambat dari versi Kubernetes dari pesawat kontrol Anda. Ini juga tidak boleh lebih dari dua versi minor lebih awal dari versi Kubernetes dari pesawat kontrol Anda. Kami menyarankan Anda menggunakan versi yang sama dengan pesawat kontrol Anda.

```
K8S_VERSION=1.31
nodes=$(kubectl get nodes -o jsonpath="{.items[?(@.status.nodeInfo.kubeletVersion==\"v$K8S_VERSION\")].metadata.name}")
for node in ${nodes[@]}
do
    echo "Tainting $node"
    kubectl taint nodes $node key=value:NoSchedule
done
```

9. Tentukan penyedia DNS klaster Anda.

```
kubectl get deployments -l k8s-app=kube-dns -n kube-system
```

Contoh output adalah sebagai berikut. Cluster ini menggunakan CoreDNS untuk resolusi DNS, tetapi klaster Anda dapat kembali sebagai gantinya): kube-dns

NAME	DESIRED	CURRENT	UP-TO-DATE	AVAILABLE	AGE
coredns	1	1	1	1	31m

10 Jika deployment Anda saat ini berjalan kurang dari dua replika, skalakan deployment menjadi dua replika. Ganti *coredns* dengan *kubedns* jika output perintah Anda sebelumnya mengembalikannya.

```
kubectl scale deployments/coredns --replicas=2 -n kube-system
```

11 Keluarkan setiap simpul yang Anda ingin hapus dari kluster Anda dengan perintah berikut:

```
kubectl drain node_name --ignore-daemonsets --delete-local-data
```

Jika Anda memutakhirkan node Anda ke versi Kubernetes baru, identifikasi dan tiriskan semua node dari versi Kubernetes tertentu (dalam hal ini, *1.31*) dengan cuplikan kode berikut.

```
K8S_VERSION=1.31
nodes=$(kubectl get nodes -o jsonpath="{.items[?(@.status.nodeInfo.kubeletVersion==\\"v$K8S_VERSION\\")].metadata.name}")
for node in ${nodes[@]}
do
    echo "Draining $node"
    kubectl drain $node --ignore-daemonsets --delete-local-data
done
```

12 Setelah node lama Anda selesai menguras, cabut aturan masuk grup keamanan yang Anda otorisasi sebelumnya. Kemudian, hapus AWS CloudFormation tumpukan untuk mengakhiri instance.

Note

Jika Anda melampirkan kebijakan IAM tambahan ke peran IAM grup node lama Anda, seperti menambahkan izin untuk [Kubernetes Cluster Autoscaler](#), lepaskan kebijakan tambahan tersebut dari peran tersebut sebelum Anda dapat menghapus tumpukan Anda. AWS CloudFormation

- a. Cabut aturan masuk yang Anda buat untuk grup keamanan node Anda sebelumnya. Dalam perintah ini, *oldNodes* adalah nama AWS CloudFormation tumpukan untuk tumpukan node lama Anda, dan *newNodes* merupakan nama tumpukan tempat Anda bermigrasi.

```
oldNodes="old_node_CFN_stack_name"
```

```
newNodes="new_node_CFN_stack_name"

oldSecGroup=$(aws cloudformation describe-stack-resources --stack-name $oldNodes \
--query 'StackResources[?
ResourceType=='AWS::EC2::SecurityGroup'].PhysicalResourceId' \
--output text)
newSecGroup=$(aws cloudformation describe-stack-resources --stack-name $newNodes \
--query 'StackResources[?
ResourceType=='AWS::EC2::SecurityGroup'].PhysicalResourceId' \
--output text)
aws ec2 revoke-security-group-ingress --group-id $oldSecGroup \
--source-group $newSecGroup --protocol -1
aws ec2 revoke-security-group-ingress --group-id $newSecGroup \
--source-group $oldSecGroup --protocol -1
```

- b. Buka [konsol AWS CloudFormation](#).
- c. Pilih tumpukan simpul lama Anda.
- d. Pilih Hapus.
- e. Di kotak dialog Hapus tumpukan konfirmasi, pilih Hapus tumpukan.

13 Edit `aws-auth` configmap untuk menghapus peran instans simpul lama dari RBAC.

```
kubectl edit configmap -n kube-system aws-auth
```

Hapus `mapRoles` entri untuk grup simpul lama.

```
apiVersion: v1
data:
  mapRoles: |
    - rolearn: arn:aws:iam::111122223333:role/nodes-1-16-NodeInstanceRole-
W70725MZQFF8
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
    - rolearn: arn:aws:iam::111122223333:role/nodes-1-15-NodeInstanceRole-
U11V27W93CX5
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes>
```

Simpan dan tutup file untuk menerapkan configmap yang diperbarui.

- 14.(Opsional) Jika Anda menggunakan [Autoscaler Klaster](#) Kubernetes, skalakan kembali deployment ke satu replika.

 Note

Anda juga harus menandai grup Auto Scaling baru Anda dengan tepat (misalnya, `k8s.io/cluster-autoscaler/enabled`, `k8s.io/cluster-autoscaler/my-cluster`) dan memperbarui perintah untuk penerapan Cluster Autoscaler Anda untuk menunjuk ke grup Auto Scaling yang baru ditandai. Untuk informasi selengkapnya, lihat [Cluster Autoscaler](#) di AWS

```
kubectl scale deployments/cluster-autoscaler --replicas=1 -n kube-system
```

- 15.(Opsional) Verifikasi bahwa Anda menggunakan versi terbaru dari [plugin Amazon VPC CNI](#) untuk Kubernetes. Anda mungkin perlu memperbarui versi CNI untuk menggunakan jenis instans terbaru yang didukung. Untuk informasi selengkapnya, lihat [the section called “Amazon VPC CNI”](#).
- 16.Jika klaster Anda menggunakan kube-dns resolusi DNS (lihat[\[migrate-determine-dns-step\]](#)), skalakan kube-dns penerapan ke satu replika.

```
kubectl scale deployments/kube-dns --replicas=1 -n kube-system
```

Perbarui tumpukan AWS CloudFormation simpul

Topik ini menjelaskan bagaimana Anda dapat memperbarui tumpukan node yang AWS CloudFormation dikelola sendiri dengan AMI baru. Anda dapat menggunakan prosedur ini untuk memperbarui node Anda ke versi baru Kubernetes setelah pembaruan klaster. Jika tidak, Anda dapat memperbarui ke AMI Amazon EKS terbaru yang dioptimalkan untuk versi Kubernetes yang ada.

 Important

Topik ini mencakup pembaruan simpul untuk simpul yang dikelola sendiri. Untuk informasi tentang penggunaan [Simplify node lifecycle dengan grup node terkelola](#), lihat [the section called “Perbarui”](#)

AWS CloudFormation Template node Amazon EKS default terbaru dikonfigurasi untuk meluncurkan instance dengan AMI baru ke dalam klaster Anda sebelum menghapus yang lama, satu per satu. Konfigurasi ini memastikan bahwa Anda selalu memiliki jumlah instans aktif yang diinginkan grup Auto Scaling di klaster selama pembaruan bergulir.

 Note

Metode ini tidak didukung untuk grup node yang dibuat dengan `eksctl`. Jika Anda membuat grup klaster atau simpul dengan `eksctl`, lihat [the section called “Migrasi”](#).

1. Tentukan penyedia DNS untuk klaster Anda.

```
kubectl get deployments -l k8s-app=kube-dns -n kube-system
```

Contoh output adalah sebagai berikut. Cluster ini menggunakan CoreDNS untuk resolusi DNS, tetapi klaster Anda mungkin kembali. `kube-dns` Output Anda mungkin terlihat berbeda tergantung pada versi `kubectl` yang Anda gunakan.

NAME	DESIRED	CURRENT	UP-TO-DATE	AVAILABLE	AGE
coredns	1	1	1	1	31m

2. Jika deployment Anda saat ini berjalan kurang dari dua replika, skalakan deployment menjadi dua replika. Ganti `coredns` dengan `kube-dns` jika output perintah Anda sebelumnya mengembalikannya.

```
kubectl scale deployments/coredns --replicas=2 -n kube-system
```

3. (Opsional) Jika Anda menggunakan Kubernetes [Cluster Autoscaler](#), turunkan skala deployment ke nol (0) replika untuk menghindari tindakan penskalaan yang bertentangan.

```
kubectl scale deployments/cluster-autoscaler --replicas=0 -n kube-system
```

4. Tentukan jenis instance dan jumlah instans yang diinginkan dari grup node Anda saat ini. Anda memasukkan nilai-nilai ini nanti ketika Anda memperbarui AWS CloudFormation template untuk grup.

a. Buka EC2 konsol Amazon di <https://console.aws.amazon.com/ec2/>.

- b. Di panel navigasi kiri, pilih Luncurkan Konfigurasi, dan catat jenis instans untuk konfigurasi peluncuran node yang ada.
 - c. Di panel navigasi kiri, pilih Grup Auto Scaling, dan catat jumlah instans yang diinginkan untuk grup Auto Scaling node yang ada.
5. Buka [konsol AWS CloudFormation](#).
 6. Pilih tumpukan grup simpul Anda, dan kemudian pilih Perbarui.
 7. Pilih Ganti template saat ini dan pilih URL Amazon S3.
 8. Untuk URL Amazon S3, rekatkan URL berikut ke area teks untuk memastikan bahwa Anda menggunakan template node AWS CloudFormation versi terbaru. Kemudian, pilih Berikutnya:

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2022-12-23/amazon-eks-nodegroup.yaml
```

9. Pada halaman Menentukan detail tumpukan, isilah parameter berikut dan pilih Selanjutnya:
 - NodeAutoScalingGroupDesiredCapacity— Masukkan jumlah instans yang diinginkan yang Anda rekam pada [langkah sebelumnya](#). Atau, masukkan jumlah node baru yang Anda inginkan untuk diskalakan saat tumpukan Anda diperbarui.
 - NodeAutoScalingGroupMaxSize— Masukkan jumlah maksimum node yang dapat diskalakan oleh grup Auto Scaling node Anda. Nilai ini harus setidaknya satu node lebih dari kapasitas yang Anda inginkan. Ini agar Anda dapat melakukan pembaruan bergulir dari node Anda tanpa mengurangi jumlah node Anda selama pembaruan.
 - NodeInstanceType— Pilih jenis instance yang Anda rekam di [langkah sebelumnya](#). Atau, pilih jenis instance yang berbeda untuk node Anda. Sebelum memilih jenis instans yang berbeda, tinjau [Pilih jenis instans EC2 node Amazon yang optimal](#). Setiap jenis EC2 instans Amazon mendukung jumlah maksimum antarmuka jaringan elastis (antarmuka jaringan) dan setiap antarmuka jaringan mendukung jumlah maksimum alamat IP. Karena setiap node pekerja dan Pod, diberi alamat IP sendiri, penting untuk memilih jenis instance yang akan mendukung jumlah maksimum Pod yang ingin Anda jalankan di setiap EC2 node Amazon. Untuk daftar jumlah antarmuka jaringan dan alamat IP yang didukung oleh jenis instans, lihat [alamat IP per antarmuka jaringan per jenis instans](#). Misalnya, tipe m5.large instance mendukung maksimal 30 alamat IP untuk node pekerja dan Pod.

 Note

[Tipe instans yang didukung untuk versi terbaru plugin Amazon VPC CNI untuk Kubernetes ditampilkan di `vpc\_ip\_resource\_limit.go on`. GitHub Anda mungkin perlu](#)

memperbarui plugin Amazon VPC CNI untuk versi Kubernetes untuk menggunakan jenis instans terbaru yang didukung. Untuk informasi selengkapnya, lihat [the section called “Amazon VPC CNI”](#).

 Important

Beberapa jenis instance mungkin tidak tersedia di semua AWS Wilayah.

- `NodelmageIdSSMParam`— Parameter Amazon EC2 Systems Manager dari ID AMI yang ingin Anda perbarui. Nilai berikut menggunakan AMI Amazon EKS terbaru yang dioptimalkan untuk versi Kubernetes. 1.33

```
/aws/service/eks/optimized-ami/1.33/amazon-linux-2/recommended/image_id
```

Anda dapat mengganti `1.33` dengan [versi Kubernetes yang didukung](#) yang sama. Atau, itu harus hingga satu versi lebih awal dari versi Kubernetes yang berjalan di bidang kontrol Anda. Kami rekomendasikan supaya Anda menyimpan simpul pada versi yang sama dengan bidang kendali Anda. Anda juga dapat mengganti `amazon-linux-2` dengan tipe AMI yang berbeda. Untuk informasi selengkapnya, lihat [the section called “Dapatkan yang terbaru IDs”](#).

 Note

Menggunakan parameter Amazon EC2 Systems Manager memungkinkan Anda memperbarui node di masa mendatang tanpa harus mencari dan menentukan ID AMI. Jika AWS CloudFormation tumpukan Anda menggunakan nilai ini, setiap pembaruan tumpukan selalu meluncurkan AMI terbaru yang dioptimalkan Amazon EKS yang dioptimalkan untuk versi Kubernetes yang Anda tentukan. Ini bahkan terjadi bahkan jika Anda tidak mengubah nilai apa pun di template.

- `NodelmageId`— Untuk menggunakan AMI kustom Anda sendiri, masukkan ID untuk AMI untuk digunakan.

 Important

Nilai ini mengesampingkan nilai apa pun yang ditentukan untuk `NodelmageldSSMParam` Jika Anda ingin menggunakan `NodelmageldSSMParam` nilainya, pastikan nilainya kosong. `Nodelmageld`

- Nonaktifkan IMDSv1 — Secara default, setiap node mendukung Layanan Metadata Instans Versi 1 (IMDSv1) dan IMDSv2. Namun, Anda dapat menonaktifkan IMDSv1. Pilih `true` jika Anda tidak ingin node atau Pod apa pun yang dijadwalkan dalam grup node untuk digunakan IMDSv1. Untuk informasi selengkapnya tentang IMDS, lihat [Mengonfigurasi layanan metadata instans](#). Jika Anda telah menerapkan peran IAM untuk akun layanan, tetapkan izin yang diperlukan secara langsung ke semua Pod yang memerlukan akses ke layanan. AWS Dengan cara ini, tidak ada Pod di kluster Anda yang memerlukan akses ke IMDS karena alasan lain, seperti mengambil Region saat ini AWS. Kemudian, Anda juga dapat menonaktifkan akses ke IMDSv2 Pod yang tidak menggunakan jaringan host. Untuk informasi selengkapnya, lihat [Membatasi akses ke profil instance yang ditetapkan ke node pekerja](#).

10.(Opsional) Di halaman Opsi, tandai sumber daya tumpukan Anda. Pilih Berikutnya.

11.Pada halaman Tinjauan, tinjau informasi Anda, nyatakan bahwa tumpukan dapat membuat sumber daya IAM, kemudian pilih Perbarui tumpukan.

 Note

Pembaruan setiap simpul dalam kluster membutuhkan waktu beberapa menit. Tunggu pembaruan semua simpul selesai sebelum melakukan langkah berikutnya.

12.Jika penyedia DNS kluster Anda `kube-dns`, skalakan `kube-dns` penerapan ke satu replika.

```
kubectl scale deployments/kube-dns --replicas=1 -n kube-system
```

13.(Opsional) Jika Anda menggunakan [Autoscaler Kluster](#) Kubernetes, skalakan kembali deployment ke jumlah replika yang Anda inginkan.

```
kubectl scale deployments/cluster-autoscaler --replicas=1 -n kube-system
```

14.(Opsional) Verifikasi bahwa Anda menggunakan versi terbaru dari [plugin Amazon VPC CNI](#) untuk Kubernetes. Anda mungkin perlu memperbarui plugin Amazon VPC CNI untuk versi Kubernetes

untuk menggunakan jenis instans terbaru yang didukung. Lihat informasi yang lebih lengkap di [the section called “Amazon VPC CNI”](#).

Sederhanakan manajemen komputasi dengan Fargate AWS

Topik ini membahas penggunaan Amazon EKS untuk menjalankan Kubernetes Pods di Fargate. AWS [Fargate adalah teknologi yang menyediakan kapasitas komputasi sesuai permintaan dan berukuran tepat untuk kontainer](#). Dengan Fargate, Anda tidak perlu menyediakan, mengonfigurasi, atau menskalakan grup mesin virtual sendiri untuk menjalankan kontainer. Anda juga tidak perlu memilih jenis server, memutuskan kapan harus menskalakan grup node Anda, atau mengoptimalkan pengemasan cluster.

[Anda dapat mengontrol Pod mana yang dimulai di Fargate dan bagaimana mereka berjalan dengan profil Fargate](#). Profil Fargate didefinisikan sebagai bagian dari cluster Amazon EKS Anda. Amazon EKS mengintegrasikan Kubernetes dengan Fargate dengan menggunakan kontroler yang dibangun menggunakan model upstream yang dapat diperluas yang disediakan AWS oleh Kubernetes. Pengontrol ini berjalan sebagai bagian dari bidang kontrol Kubernetes yang dikelola Amazon EKS dan bertanggung jawab untuk menjadwalkan Pod Kubernetes asli ke Fargate. Pengendali Fargate termasuk penjadwal baru yang berjalan dengan penjadwal Kubernetes default selain beberapa pengendali masuk urusan mutasi dan validasi. Ketika Anda memulai sebuah Pod yang memenuhi kriteria untuk berjalan di Fargate, pengendali Fargate yang berjalan di klaster mengenali, memperbarui, dan menjadwalkan Pod ke Fargate.

Topik ini menjelaskan berbagai komponen Pod yang berjalan di Fargate, dan memunculkan pertimbangan khusus untuk menggunakan Fargate dengan Amazon EKS.

AWS Pertimbangan Fargate

Berikut adalah beberapa hal yang perlu dipertimbangkan tentang menggunakan Fargate di Amazon EKS.

- Setiap Pod yang berjalan di Fargate memiliki batas komputasinya sendiri. Mereka tidak berbagi kernel yang mendasarinya, sumber daya CPU, sumber daya memori, atau elastic network interface dengan Pod lain.
- Network Load Balancers dan Application Load Balancers (ALBs) dapat digunakan dengan Fargate dengan target IP saja. Untuk informasi selengkapnya, lihat [the section called “Buat penyeimbang beban jaringan”](#) dan [the section called “Penyeimbangan beban aplikasi”](#).

- Layanan terbuka Fargate hanya berjalan pada mode IP tipe target, dan bukan pada mode IP node. Cara yang disarankan untuk memeriksa konektivitas dari layanan yang berjalan pada node terkelola dan layanan yang berjalan di Fargate adalah dengan menghubungkan melalui nama layanan.
- Pod harus cocok dengan profil Fargate pada saat mereka dijadwalkan untuk berjalan di Fargate. Pod yang tidak cocok dengan profil Fargate mungkin macet sebagai Pending. Jika ada profil Fargate yang cocok, Anda dapat menghapus Pod yang tertunda yang telah Anda buat untuk menjadwalkannya kembali ke Fargate.
- Daemonset tidak didukung di Fargate. Jika aplikasi Anda memerlukan daemon, konfigurasi ulang daemon tersebut untuk dijalankan sebagai wadah sespan di Pod Anda.
- Kontainer istimewa tidak didukung di Fargate.
- Pod yang berjalan di Fargate tidak dapat menentukan HostPort atau HostNetwork dalam manifes Pod.
- Default `nofile` dan `nproc` soft limit adalah 1024 dan hard limit adalah 65535 untuk Fargate Pods.
- GPUs saat ini tidak tersedia di Fargate.
- Pod yang berjalan di Fargate hanya didukung pada subnet pribadi (dengan akses gateway NAT ke AWS layanan, tetapi bukan rute langsung ke Internet Gateway), jadi VPC kluster Anda harus memiliki subnet pribadi yang tersedia. Untuk kluster tanpa akses internet luar, lihat [the section called “kluster privat”](#).
- Anda dapat menggunakan [sumber daya Adjust pod dengan Vertical Pod Autoscaler](#) untuk mengatur ukuran CPU dan memori awal yang benar untuk Pod Fargate Anda, dan kemudian menggunakan [penerapan pod Scale dengan Horizontal Pod Autoscaler untuk menskalakan Pod](#) tersebut. Jika Anda ingin Vertical Pod Autoscaler untuk secara otomatis men-deploy ulang Pod ke Fargate dengan kombinasi CPU dan memori yang lebih besar, atur mode untuk Vertical Pod Autoscaler ke salah satu atau untuk memastikan fungsionalitas yang benar. Auto Recreate Untuk informasi selengkapnya, lihat dokumentasi [Vertical Pod Autoscaler](#) pada GitHub.
- Resolusi DNS dan nama host DNS harus diaktifkan untuk VPC Anda. Untuk informasi selengkapnya, lihat [Melihat dan memperbarui dukungan DNS untuk VPC Anda](#).
- Amazon EKS Fargate menambahkan aplikasi defense-in-depth Kubernetes dengan mengisolasi setiap Pod dalam Virtual Machine (VM). Batas VM ini mencegah akses ke sumber daya berbasis host yang digunakan oleh Pod lain jika terjadi pelarian kontainer, yang merupakan metode umum untuk menyerang aplikasi kontainer dan mendapatkan akses ke sumber daya di luar container.

Menggunakan Amazon EKS tidak mengubah tanggung jawab Anda berdasarkan [model tanggung jawab bersama](#). Anda harus hati-hati mempertimbangkan konfigurasi keamanan kluster dan kontrol

tata kelola. Cara teraman untuk mengisolasi aplikasi adalah selalu menjalankannya di cluster terpisah.

- Profil Fargate mendukung penentuan subnet dari blok CIDR sekunder VPC. Anda mungkin ingin menentukan blok CIDR sekunder. Ini karena ada sejumlah alamat IP yang tersedia di subnet. Akibatnya, ada juga sejumlah Pod yang dapat dibuat di kluster. Dengan menggunakan subnet yang berbeda untuk Pod, Anda dapat menambah jumlah alamat IP yang tersedia. Untuk informasi selengkapnya, lihat [Menambahkan blok IPv4 CIDR ke VPC](#).
- Layanan metadata EC2 instans Amazon (IMDS) tidak tersedia untuk Pod yang di-deploy ke node Fargate. [Jika Anda memiliki Pod yang di-deploy ke Fargate yang membutuhkan kredensial IAM, tetapkan Pod tersebut ke Pod Anda menggunakan peran IAM untuk akun layanan](#). Jika Pod Anda memerlukan akses ke informasi lain yang tersedia melalui IMDS, maka Anda harus membuat kode keras informasi ini ke dalam spesifikasi Pod Anda. Ini termasuk AWS Region atau Availability Zone tempat Pod digunakan.
- Anda tidak dapat menerapkan Pod Fargate ke AWS Outposts, AWS Wavelength, atau Local Zones. AWS
- Amazon EKS harus menambal Pod Fargate secara berkala agar tetap aman. Kami mencoba pembaruan dengan cara yang mengurangi dampak, tetapi ada kalanya Pod harus dihapus jika tidak berhasil diusir. Ada beberapa tindakan yang dapat Anda lakukan untuk meminimalkan gangguan. Untuk informasi selengkapnya, lihat [the section called “Acara penambalan OS”](#).
- [Plugin Amazon VPC CNI untuk Amazon EKS diinstal](#) pada node Fargate. Anda tidak dapat menggunakan [plugin Alternatif CNI untuk kluster Amazon EKS dengan node Fargate](#).
- Pod yang berjalan di Fargate secara otomatis memasang sistem file Amazon EFS, tanpa memerlukan langkah penginstalan driver secara manual. Anda tidak dapat menggunakan penyediaan volume persisten dinamis dengan node Fargate, tetapi Anda dapat menggunakan penyediaan statis.
- Amazon EKS tidak mendukung Fargate Spot.
- Anda tidak dapat memasang volume Amazon EBS ke Pod Fargate.
- Anda dapat menjalankan pengontrol Amazon EBS CSI di node Fargate, tetapi node Amazon EBS CSI DaemonSet hanya dapat berjalan di instans Amazon. EC2
- Setelah [Job Kubernetes](#) Completed ditandai Failed atau, Pod yang dibuat oleh Job biasanya akan tetap ada. Perilaku ini memungkinkan Anda untuk melihat log dan hasil Anda, tetapi dengan Fargate Anda akan dikenakan biaya jika Anda tidak membersihkan Job sesudahnya.

Untuk menghapus Pod terkait secara otomatis setelah Job selesai atau gagal, Anda dapat menentukan periode waktu menggunakan pengontrol time-to-live (TTL). Contoh berikut menunjukkan spesifikasi `.spec.ttlSecondsAfterFinished` dalam manifes Job Anda.

```
apiVersion: batch/v1
kind: Job
metadata:
  name: busybox
spec:
  template:
    spec:
      containers:
      - name: busybox
        image: busybox
        command: ["/bin/sh", "-c", "sleep 10"]
        restartPolicy: Never
      ttlSecondsAfterFinished: 60 # <-- TTL controller
```

Tabel Perbandingan Fargate

Kriteria	AWS Fargate
Dapat dikerahkan ke Outposts AWS	Tidak
Dapat dikerahkan ke Zona AWS Lokal	Tidak
Dapat menjalankan kontainer yang memerlukan Windows	Tidak
Dapat menjalankan kontainer yang membutuhkan Linux	Ya
Dapat menjalankan beban kerja yang memerlukan Chip Inferentia	Tidak
Dapat menjalankan beban kerja yang memerlukan GPU	Tidak

Kriteria	AWS Fargate
Dapat menjalankan beban kerja yang memerlukan prosesor Arm	Tidak
Dapat menjalankan AWS Bottlerocket	Tidak
Pod berbagi lingkungan runtime kernel dengan Pod lain	Tidak - Setiap Pod memiliki kernel khusus
Pod berbagi CPU, memori, penyimpanan, dan sumber daya jaringan dengan Pod lain.	Tidak - Setiap Pod memiliki sumber daya khusus dan dapat diukur secara independen untuk memaksimalkan pemanfaatan sumber daya.
Pod dapat menggunakan lebih banyak perangkat keras dan memori daripada yang diminta dalam spesifikasi Pod	Tidak - Pod dapat di-deploy ulang menggunakan vCPU dan konfigurasi memori yang lebih besar.
Harus menerapkan dan mengelola instans Amazon EC2	Tidak
Harus mengamankan, memelihara, dan menambal sistem operasi EC2 instans Amazon	Tidak
Dapat memberikan argumen bootstrap pada penerapan node, seperti argumen kubelet ekstra.	Tidak
Dapat menetapkan alamat IP ke Pod dari blok CIDR yang berbeda dari alamat IP yang ditetapkan ke node.	Tidak
Dapat SSH ke simpul	Tidak - Tidak ada sistem operasi host node untuk SSH.
Dapat men-deploy AMI kustom Anda sendiri ke simpul	Tidak

Kriteria	AWS Fargate
Dapat men-deploy CNI kustom Anda sendiri ke simpul	Tidak
Harus memperbarui node AMI sendiri	Tidak
Harus memperbarui versi node Kubernetes sendiri	Tidak - Anda tidak mengelola node.
Dapat menggunakan penyimpanan Amazon EBS dengan Pod	Tidak
Dapat menggunakan penyimpanan Amazon EFS dengan Pod	Ya
Dapat menggunakan Amazon FSx untuk penyimpanan Lustre dengan Pod	Tidak
Dapat menggunakan Network Load Balancer untuk layanan	Ya, saat menggunakan Buat penyeimbang beban jaringan
Pods dapat berjalan dalam subnet publik	Tidak
Dapat menetapkan grup keamanan VPC yang berbeda ke masing-masing Pod	Ya
Dapat menjalankan Kubernetes DaemonSets	Tidak
Support HostPort dan HostNetwork dalam manifes Pod	Tidak
AWS Ketersediaan wilayah	Beberapa wilayah yang didukung Amazon EKS
Dapat menjalankan kontainer di host EC2 khusus Amazon	Tidak

Kriteria	AWS Fargate
Harga	Biaya memori Fargate secara individu dan konfigurasi CPU. Setiap Pod memiliki biaya sendiri. Untuk informasi lebih lanjut, lihat AWS harga Fargate .

Memulai AWS Fargate untuk klaster Anda

Topik ini menjelaskan cara memulai menjalankan Pod di AWS Fargate dengan klaster Amazon EKS Anda.

Jika Anda membatasi akses ke titik akhir publik klaster Anda menggunakan blok CIDR, sebaiknya Anda juga mengaktifkan akses titik akhir pribadi. Dengan cara ini, Fargate Pod dapat berkomunikasi dengan cluster. Tanpa mengaktifkan titik akhir pribadi, blok CIDR yang Anda tentukan untuk akses publik harus menyertakan sumber keluar dari VPC Anda. Untuk informasi selengkapnya, lihat [the section called “Akses titik akhir klaster”](#).

Prasyarat

Sebuah klaster yang sudah ada. Jika Anda belum memiliki kluster Amazon EKS, lihat [Memulai](#).

Langkah 1: Pastikan node yang ada dapat berkomunikasi dengan Fargate Pods

Jika Anda bekerja dengan cluster baru tanpa node, atau cluster dengan hanya grup node terkelola (lihat [the section called “Grup simpul terkelola”](#)), Anda dapat melompat ke [the section called “Langkah 2: Buat peran eksekusi Fargate Pod”](#).

Asumsikan bahwa Anda bekerja dengan cluster yang sudah ada yang sudah memiliki node yang terkait dengannya. Pastikan bahwa Pod pada node ini dapat berkomunikasi secara bebas dengan Pod yang berjalan di Fargate. Pod yang berjalan di Fargate secara otomatis dikonfigurasi untuk menggunakan grup keamanan klaster untuk klaster yang terkait dengannya. Pastikan bahwa setiap node yang ada di cluster Anda dapat mengirim dan menerima lalu lintas ke dan dari grup keamanan klaster. Grup node terkelola secara otomatis dikonfigurasi untuk menggunakan grup keamanan klaster juga, jadi Anda tidak perlu memodifikasi atau memeriksa kompatibilitas ini (lihat [the section called “Grup simpul terkelola”](#)).

Untuk grup node yang sudah ada yang dibuat dengan eksctl atau AWS CloudFormation templat terkelola Amazon EKS, Anda dapat menambahkan grup keamanan klaster ke node secara manual.

Atau, sebagai alternatif, Anda dapat memodifikasi templat peluncuran grup Auto Scaling untuk grup node untuk melampirkan grup keamanan klaster ke instance. Untuk informasi selengkapnya, lihat [Mengubah grup keamanan instans](#) di Panduan Pengguna Amazon VPC.

Anda dapat memeriksa grup keamanan untuk klaster Anda di bagian AWS Management Console bawah Jaringan untuk klaster. Atau, Anda dapat melakukan ini menggunakan perintah AWS CLI berikut. Saat menggunakan perintah ini, ganti `<my-cluster>` dengan nama cluster Anda.

```
aws eks describe-cluster --name <my-cluster> --query  
cluster.resourcesVpcConfig.clusterSecurityGroupId
```

Langkah 2: Buat peran eksekusi Fargate Pod

Saat klaster Anda membuat Pod di AWS Fargate, komponen yang berjalan di infrastruktur Fargate harus melakukan panggilan atas nama Anda. AWS APIs Peran eksekusi Amazon EKS Pod memberikan izin IAM untuk melakukan ini. Untuk membuat peran eksekusi AWS Fargate Pod, lihat [the section called “Peran IAM eksekusi pod”](#)

Note

Jika Anda membuat klaster dengan `eksctl` menggunakan `--fargate` opsi, klaster Anda sudah memiliki peran eksekusi Pod yang dapat Anda temukan di konsol IAM dengan pola `eksctl-my-cluster-FargatePodExecutionRole-ABCDEFGHIJKL` tersebut. Demikian pula, jika Anda gunakan `eksctl` untuk membuat profil Fargate Anda, `eksctl` buat peran eksekusi Pod Anda jika belum dibuat.

Langkah 3: Buat profil Fargate untuk cluster Anda

Sebelum Anda dapat menjadwalkan Pod yang berjalan di Fargate di klaster Anda, Anda harus menentukan profil Fargate yang menentukan Pod mana yang menggunakan Fargate saat diluncurkan. Untuk informasi selengkapnya, lihat [the section called “Tentukan profil”](#).

Note

Jika Anda membuat klaster dengan `eksctl` menggunakan `--fargate` opsi, maka profil Fargate sudah dibuat untuk klaster Anda dengan pemilih untuk semua Pod di dan

namespace. kube-system default Gunakan prosedur berikut untuk membuat profil Fargate untuk namespaces lain yang ingin Anda gunakan dengan Fargate.

Anda dapat membuat profil Fargate menggunakan salah satu alat ini:

- [the section called “eksctl”](#)
- [the section called “AWS Management Console”](#)

eksctl

Prosedur ini membutuhkan eksctl versi 0.210.0 atau yang lebih baru. Anda dapat memeriksa versi Anda dengan perintah berikut:

```
eksctl version
```

Untuk petunjuk tentang cara menginstal atau meningkatkan eksctl, lihat [Instalasi](#) dalam eksctl dokumentasi.

Untuk membuat profil Fargate dengan **eksctl**

Buat profil Fargate Anda dengan eksctl perintah berikut, ganti masing-masing <example value> dengan nilai Anda sendiri. Anda diminta untuk menentukan namespace. Namun, --labels opsi tidak diperlukan.

```
eksctl create fargateprofile \  
  --cluster <my-cluster> \  
  --name <my-fargate-profile> \  
  --namespace <my-kubernetes-namespace> \  
  --labels <key=value>
```

Anda dapat menggunakan wildcard tertentu untuk <my-kubernetes-namespace> dan <key=value> label. Untuk informasi selengkapnya, lihat [the section called “Wildcard profil Fargate”](#).

AWS Management Console

Untuk membuat profil Fargate dengan AWS Management Console

1. Buka [konsol Amazon EKS](#).

2. Pilih klaster untuk membuat profil Fargate untuk.
3. Pilih tab Compute.
4. Di bawah Profil Fargate, pilih Tambahkan profil Fargate.
5. Pada halaman profil Konfigurasi Fargate, lakukan hal berikut:
 - a. Untuk Nama, masukkan nama untuk profil Fargate Anda. Nama harus unik.
 - b. Untuk peran eksekusi Pod, pilih peran eksekusi Pod yang akan digunakan dengan profil Fargate Anda. Hanya peran IAM dengan kepala `eks-fargate-pods.amazonaws.com` layanan yang ditampilkan. Jika Anda tidak melihat peran apa pun yang terdaftar, Anda harus membuatnya. Untuk informasi selengkapnya, lihat [the section called “Peran IAM eksekusi pod”](#).
 - c. Ubah Subnet yang dipilih sesuai kebutuhan.

**Note**

Hanya subnet pribadi yang didukung untuk Pod yang berjalan di Fargate.

- d. Untuk Tandai, Anda dapat menandai profil Fargate Anda secara opsional. Tag ini tidak menyebar ke sumber daya lain yang terkait dengan profil seperti Pod.
 - e. Pilih Berikutnya.
6. Pada halaman pemilihan Configure Pod, lakukan hal berikut:
- a. Untuk Namespace, masukkan namespace yang cocok dengan Pod.
 - Anda dapat menggunakan ruang nama tertentu untuk mencocokkan, seperti `kube-system` atau `default`
 - Anda dapat menggunakan wildcard tertentu (misalnya, `prod-*`) untuk mencocokkan beberapa ruang nama (misalnya, `prod-deployment` dan `prod-test`). Untuk informasi selengkapnya, lihat [the section called “Wildcard profil Fargate”](#).
 - b. (Opsional) Tambahkan label Kubernetes ke pemilih. Secara khusus menambahnya ke salah satu yang harus dicocokkan dengan Pod di namespace tertentu.
 - Anda dapat menambahkan label `infrastructure: fargate` ke pemilih sehingga hanya Pod di namespace tertentu yang juga memiliki label `infrastructure: fargate` Kubernetes yang cocok dengan pemilih.
 - Anda dapat menggunakan wildcard tertentu (misalnya, `key?: value?`) untuk mencocokkan beberapa ruang nama (misalnya, `keya: valuea` dan `keyb: valueb`). Untuk informasi selengkapnya, lihat [the section called “Wildcard profil Fargate”](#).
 - c. Pilih Berikutnya.

7. Pada halaman Periksa dan buat, tinjau informasi untuk profil Fargate Anda dan pilih Buat.

Langkah 4: Perbarui CoreDNS

Secara default, CoreDNS dikonfigurasi untuk berjalan di EC2 infrastruktur Amazon di kluster Amazon EKS. Jika Anda hanya ingin menjalankan Pod di Fargate di kluster Anda, selesaikan langkah-langkah berikut.

Note

Jika Anda membuat cluster Anda dengan eksctl menggunakan `--fargate` opsi, maka Anda dapat melompat ke [the section called “Langkah selanjutnya”](#).

1. Buat profil Fargate untuk CoreDNS dengan perintah berikut. Ganti `<my-cluster>` dengan nama kluster Anda, `<111122223333>` dengan ID akun Anda, `<AmazonEKSFargatePodExecutionRole>` dengan nama peran eksekusi Pod Anda, dan `<0000000000000000a>`, `<0000000000000000b>`, dan `<0000000000000000c>` dengan subnet pribadi Anda. IDs Jika Anda tidak memiliki peran eksekusi Pod, Anda harus membuatnya terlebih dahulu (lihat [the section called “Langkah 2: Buat peran eksekusi Fargate Pod”](#)).

Important

Peran ARN tidak dapat menyertakan [jalur](#) selain `/`. Misalnya, jika nama peran `Andadevelopment/apps/AmazonEKSFargatePodExecutionRole`, Anda perlu mengubahnya menjadi `AmazonEKSFargatePodExecutionRole` saat menentukan ARN untuk peran tersebut. Format ARN peran harus `arn:aws:iam::<111122223333>:role/<AmazonEKSFargatePodExecutionRole>`.

```
aws eks create-fargate-profile \
  --fargate-profile-name coredns \
  --cluster-name <my-cluster> \
  --pod-execution-role-arn arn:aws:iam::<111122223333>:role/
  <AmazonEKSFargatePodExecutionRole> \
  --selectors namespace=kube-system,labels={k8s-app=kube-dns} \
  --subnets subnet-<0000000000000000a> subnet-<0000000000000000b> subnet-
  <0000000000000000c>
```

2. Memicu peluncuran penyebaran. coredns

```
kubectl rollout restart -n kube-system deployment coredns
```

Langkah selanjutnya

- Anda dapat mulai memigrasi aplikasi yang ada untuk berjalan di Fargate dengan alur kerja berikut.
 - a. [the section called “Buat profil Fargate”](#) yang cocok dengan namespace Kubernetes dan label Kubernetes aplikasi Anda.
 - b. Hapus dan buat ulang Pod yang ada sehingga dijadwalkan di Fargate. Ubah <namespace> dan <deployment-type> untuk memperbarui Pod spesifik Anda.

```
kubectl rollout restart -n <namespace> deployment <deployment-type>
```

- Terapkan [the section called “Penyeimbangan beban aplikasi”](#) untuk mengizinkan objek Ingress untuk Pod Anda yang berjalan di Fargate.
- Anda dapat menggunakannya [the section called “Vertical Pod Autoscaler”](#) untuk mengatur ukuran CPU dan memori awal yang benar untuk Pod Fargate Anda, lalu gunakan [the section called “Penskala Otomatis Pod Horizontal”](#) untuk menskalakan Pod tersebut. Jika Anda ingin Vertical Pod Autoscaler untuk secara otomatis men-deploy ulang Pod ke Fargate dengan kombinasi CPU dan memori yang lebih tinggi, atur mode Vertical Pod Autoscaler ke salah satu atau. Auto Recreate Ini untuk memastikan fungsionalitas yang benar. Untuk informasi selengkapnya, lihat dokumentasi [Vertical Pod Autoscaler](#) pada. GitHub
- Anda dapat mengatur kolektor [AWS Distro for OpenTelemetry](#) (ADOT) untuk pemantauan aplikasi dengan mengikuti petunjuk [ini](#).

Tentukan Pod mana yang menggunakan AWS Fargate saat diluncurkan

Sebelum Anda menjadwalkan Pod di Fargate di kluster Anda, Anda harus menentukan setidaknya satu profil Fargate yang menentukan Pod mana yang menggunakan Fargate saat diluncurkan.

Sebagai administrator, Anda dapat menggunakan profil Fargate untuk mendeklarasikan Pod mana yang berjalan di Fargate. Anda dapat melakukan ini melalui pemilih profil. Anda dapat menambahkan hingga lima pemilih ke setiap profil. Setiap pemilih harus berisi namespace. Pemilih juga dapat menyertakan label. Bidang label terdiri dari beberapa pasangan nilai kunci opsional. Pod yang cocok dengan pemilih dijadwalkan di Fargate. Pod dicocokkan menggunakan namespace dan label yang

ditentukan dalam pemilih. Jika pemilih namespace didefinisikan tanpa label, Amazon EKS mencoba menjadwalkan semua Pod yang berjalan di namespace tersebut ke Fargate menggunakan profil tersebut. Jika to-be-scheduled Pod cocok dengan salah satu pemilih di profil Fargate, maka Pod tersebut dijadwalkan di Fargate.

Jika sebuah Pod cocok dengan beberapa profil Fargate, Anda dapat menentukan profil mana yang digunakan Pod dengan menambahkan label Kubernetes berikut ke spesifikasi Pod: `eks.amazonaws.com/fargate-profile: my-fargate-profile` Pod harus cocok dengan pemilih dalam profil tersebut untuk dijadwalkan ke Fargate. Aturan afinitas/anti-afinitas Kubernetes tidak berlaku dan tidak diperlukan dengan Pod Fargate Amazon EKS.

Saat Anda membuat profil Fargate, Anda harus menentukan peran eksekusi Pod. Peran eksekusi ini untuk komponen Amazon EKS yang berjalan di infrastruktur Fargate menggunakan profil. Ini ditambahkan ke Kubernetes [Role Based Access Control](#) (RBAC) cluster untuk otorisasi. Dengan begitu, kubelet yang berjalan pada infrastruktur Fargate dapat mendaftar dengan cluster Amazon EKS Anda dan muncul di cluster Anda sebagai node. Peran eksekusi Pod juga memberikan izin IAM ke infrastruktur Fargate untuk memungkinkan akses baca ke repositori image Amazon ECR. Untuk informasi selengkapnya, lihat [the section called “Peran IAM eksekusi pod”](#).

Profil Fargate tidak dapat diubah. Namun, Anda dapat membuat profil baru yang diperbarui untuk mengganti profil yang ada, dan kemudian menghapus yang asli.

 Note

Pod apa pun yang berjalan menggunakan profil Fargate akan dihentikan dan dimasukkan ke dalam status tertunda saat profil dihapus.

Jika ada profil Fargate dalam kluster dalam DELETING status, Anda harus menunggu sampai setelah profil Fargate dihapus sebelum Anda membuat profil lain di cluster itu.

 Note

Fargate saat ini tidak mendukung Kubernetes. [topologySpreadConstraints](#)

Amazon EKS dan Fargate menyebarkan Pod di setiap subnet yang ditentukan dalam profil Fargate. Namun, Anda mungkin berakhir dengan penyebaran yang tidak merata. Jika Anda harus memiliki spread yang merata, gunakan dua profil Fargate. Bahkan spread penting dalam skenario di mana

Anda ingin menerapkan dua replika dan tidak ingin downtime. Kami merekomendasikan bahwa setiap profil hanya memiliki satu subnet.

Komponen profil Fargate

Komponen-komponen berikut yang terkandung dalam profil Fargate.

Peran eksekusi pod

Saat klaster Anda membuat Pod di AWS Fargate, kubelet yang berjalan di infrastruktur Fargate harus melakukan panggilan atas nama Anda. AWS APIs Misalnya, perlu melakukan panggilan untuk menarik gambar kontainer dari Amazon ECR. Peran eksekusi Amazon EKS Pod memberikan izin IAM untuk melakukan ini.

Saat Anda membuat profil Fargate, Anda harus menentukan peran eksekusi Pod yang akan digunakan dengan Pod Anda. Peran ini ditambahkan ke Kubernetes [Role-based access control](#) (RBAC) klaster untuk otorisasi. Ini agar yang berjalan di infrastruktur Fargate dapat mendaftar dengan cluster Amazon EKS Anda dan muncul di cluster Anda sebagai node. kubelet Untuk informasi selengkapnya, lihat [the section called “Peran IAM eksekusi pod”](#).

Subnet

Subnet untuk meluncurkan Pod ke dalamnya menggunakan profil ini. IDs Saat ini, Pod yang berjalan di Fargate tidak diberi alamat IP publik. Oleh karena itu, hanya subnet pribadi tanpa rute langsung ke Internet Gateway yang diterima untuk parameter ini.

Selektor

Selector untuk mencocokkan Pod untuk menggunakan profil Fargate ini. Anda dapat menentukan hingga lima pemilih dalam profil Fargate. Penyeleksi memiliki komponen-komponen berikut:

- **Namespace** – Anda harus menentukan namespace untuk pemilih. Selector hanya cocok dengan Pod yang dibuat di namespace ini. Namun, Anda dapat membuat beberapa penyeleksi untuk menargetkan beberapa ruang nama.
- **Label** — Anda dapat menentukan label Kubernetes untuk dicocokkan dengan pemilih. Pemilih hanya cocok dengan Pod yang memiliki semua label yang ditentukan dalam pemilih.

Wildcard profil Fargate

Selain karakter yang diizinkan oleh Kubernetes, Anda diizinkan untuk menggunakan * dan ? dalam kriteria pemilih untuk namespace, kunci label, dan nilai label:

- *mewakili tidak ada, satu, atau beberapa karakter. Misalnya, `prod*` dapat mewakili `prod` dan `prod-metrics`.
- ?mewakili satu karakter (misalnya, `value?` dapat mewakili `valuea`). Namun, itu tidak dapat mewakili `value` dan `value-a`, karena hanya ? dapat mewakili tepat satu karakter.

Karakter wildcard ini dapat digunakan dalam posisi apa pun dan dalam kombinasi (misalnya, `prod**dev`, dan `frontend*?`). Wildcard lain dan bentuk pencocokan pola, seperti ekspresi reguler, tidak didukung.

Jika ada beberapa profil yang cocok untuk namespace dan label dalam spesifikasi Pod, Fargate mengambil profil berdasarkan pengurutan alfanumerik berdasarkan nama profil. Misalnya, jika kedua profil A (dengan nama `beta-workload`) dan profil B (dengan nama `prod-workload`) memiliki pemilih yang cocok untuk Pod yang akan diluncurkan, Fargate memilih profil A `beta-workload` () untuk Pod. Pod memiliki label dengan profil A pada Pod (misalnya, `eks.amazonaws.com/fargate-profile=beta-workload`).

Jika Anda ingin memigrasikan Pod Fargate yang ada ke profil baru yang menggunakan wildcard, ada dua cara untuk melakukannya:

- Buat profil baru dengan penyeleksi yang cocok, lalu hapus profil lama. Pod yang diberi label dengan profil lama dijadwal ulang ke profil baru yang cocok.
- Jika Anda ingin memigrasikan beban kerja tetapi tidak yakin label Fargate pada setiap Pod Fargate, Anda dapat menggunakan metode berikut. Buat profil baru dengan nama yang mengurutkan secara alfanumerik terlebih dahulu di antara profil di cluster yang sama. Kemudian, daur ulang Pod Fargate yang perlu dimigrasikan ke profil baru.

Buat profil Fargate

Bagian ini menjelaskan cara membuat profil Fargate. Anda juga harus telah membuat peran eksekusi Pod untuk digunakan untuk profil Fargate Anda. Untuk informasi selengkapnya, lihat [the section called “Peran IAM eksekusi pod”](#). Pod yang berjalan di Fargate hanya didukung pada subnet pribadi dengan akses [gateway NAT](#) ke AWS layanan, tetapi bukan rute langsung ke Internet Gateway. Ini agar VPC cluster Anda harus memiliki subnet pribadi yang tersedia.

Anda dapat membuat profil dengan yang berikut ini:

- [the section called “eksctl”](#)

- [the section called “AWS Management Console”](#)

eksctl

Untuk membuat profil Fargate dengan **eksctl**

Buat profil Fargate Anda dengan **eksctl** perintah berikut, ganti setiap nilai contoh dengan nilai Anda sendiri. Anda diminta untuk menentukan namespace. Namun, `--labels` opsi tidak diperlukan.

```
eksctl create fargateprofile \  
  --cluster my-cluster \  
  --name my-fargate-profile \  
  --namespace my-kubernetes-namespace \  
  --labels key=value
```

Anda dapat menggunakan wildcard tertentu untuk `my-kubernetes-namespace` dan `key=value` label. Untuk informasi selengkapnya, lihat [the section called “Wildcard profil Fargate”](#).

AWS Management Console

Untuk membuat profil Fargate dengan AWS Management Console

1. Buka [konsol Amazon EKS](#).
2. Pilih klaster untuk membuat profil Fargate untuk.
3. Pilih tab Compute.
4. Di bawah Profil Fargate, pilih Tambahkan profil Fargate.
5. Pada halaman profil Konfigurasi Fargate, lakukan hal berikut:
 - a. Untuk Nama, masukkan nama unik untuk profil Fargate Anda, seperti `my-profile`.
 - b. Untuk peran eksekusi Pod, pilih peran eksekusi Pod yang akan digunakan dengan profil Fargate Anda. Hanya peran IAM dengan kepala `eks-fargate-pods.amazonaws.com` layanan yang ditampilkan. Jika Anda tidak melihat peran apa pun yang terdaftar, Anda harus membuatnya. Untuk informasi selengkapnya, lihat [the section called “Peran IAM eksekusi pod”](#).
 - c. Ubah Subnet yang dipilih sesuai kebutuhan.

Note

Hanya subnet pribadi yang didukung untuk Pod yang berjalan di Fargate.

- d. Untuk Tandai, Anda dapat menandai profil Fargate Anda secara opsional. Tag ini tidak menyebar ke sumber daya lain yang terkait dengan profil, seperti Pod.
 - e. Pilih Berikutnya.
6. Pada halaman pemilihan Configure Pod, lakukan hal berikut:
- a. Untuk Namespace, masukkan namespace yang cocok dengan Pod.
 - Anda dapat menggunakan ruang nama tertentu untuk mencocokkan, seperti kube-system atau default
 - Anda dapat menggunakan wildcard tertentu (misalnya, prod-*) untuk mencocokkan beberapa ruang nama (misalnya, prod-deployment dan). prod-test Untuk informasi selengkapnya, lihat [the section called “Wildcard profil Fargate”](#).
 - b. (Opsional) Tambahkan label Kubernetes ke pemilih. Secara khusus, tambahkan ke salah satu yang harus dicocokkan dengan Pod di namespace tertentu.
 - Anda dapat menambahkan label infrastructure: fargate ke pemilih sehingga hanya Pod di namespace tertentu yang juga memiliki label infrastructure: fargate Kubernetes yang cocok dengan pemilih.
 - Anda dapat menggunakan wildcard tertentu (misalnya, key?: value?) untuk mencocokkan beberapa ruang nama (misalnya, keya: valuea dan). keyb: valueb Untuk informasi selengkapnya, lihat [the section called “Wildcard profil Fargate”](#).
 - c. Pilih Berikutnya.
7. Pada halaman Periksa dan buat, tinjau informasi untuk profil Fargate Anda dan pilih Buat.

Hapus profil Fargate

Topik ini menjelaskan cara menghapus profil Fargate. Saat Anda menghapus profil Fargate, semua Pod yang dijadwalkan ke Fargate dengan profil akan dihapus. Jika Pod tersebut cocok dengan profil Fargate lainnya, maka mereka dijadwalkan di Fargate dengan profil itu. Jika mereka tidak lagi cocok dengan profil Fargate, maka mereka tidak dijadwalkan ke Fargate dan mungkin tetap tertunda.

Hanya satu profil Fargate dalam sebuah kluster yang dapat memiliki status DELETING pada satu waktu. Tunggu profil Fargate selesai dihapus sebelum Anda dapat menghapus profil lain di cluster itu.

Anda dapat menghapus profil dengan salah satu alat berikut:

- [the section called “eksctl”](#)
- [the section called “AWS Management Console”](#)

- [the section called “AWS CLI”](#)

eksctl

Hapus profil Fargate dengan **eksctl**

Gunakan perintah berikut untuk menghapus profil dari klaster. Ganti setiap *example value* dengan nilai-nilai Anda sendiri.

```
eksctl delete fargateprofile --name my-profile --cluster my-cluster
```

AWS Management Console

Hapus profil Fargate dengan AWS Management Console

1. Buka [konsol Amazon EKS](#).
2. Pada panel navigasi sebelah kiri, pilih Klaster. Dalam daftar cluster, pilih cluster tempat Anda ingin menghapus profil Fargate.
3. Pilih tab Compute.
4. Pilih profil Fargate untuk dihapus, lalu pilih Hapus.
5. Pada halaman Hapus profil Fargate, masukkan nama profil, lalu pilih Hapus.

AWS CLI

Hapus profil Fargate dengan CLI AWS

Gunakan perintah berikut untuk menghapus profil dari klaster. Ganti setiap *example value* dengan nilai-nilai Anda sendiri.

```
aws eks delete-fargate-profile --fargate-profile-name my-profile --cluster-name my-cluster
```

Memahami detail konfigurasi Fargate Pod

Bagian ini menjelaskan beberapa detail konfigurasi Pod yang unik untuk menjalankan Pod Kubernetes di Fargate. AWS

Pod CPU dan memori

Dengan Kubernetes, Anda dapat menentukan permintaan, jumlah minimum vCPU, dan sumber daya memori yang dialokasikan ke setiap kontainer dalam sebuah Pod. Pod dijadwalkan oleh Kubernetes untuk memastikan bahwa setidaknya sumber daya yang diminta untuk setiap Pod tersedia di sumber daya komputasi. Untuk informasi lebih lanjut, lihat [Mengelola sumber daya komputasi untuk kontainer](#) dalam dokumentasi Kubernetes.

Note

Karena Amazon EKS Fargate hanya menjalankan satu Pod per node, skenario penggusuran Pod jika sumber daya lebih sedikit tidak terjadi. Semua Pod Amazon EKS Fargate berjalan dengan prioritas terjamin, sehingga CPU dan memori yang diminta harus sama dengan batas untuk semua container. Untuk informasi selengkapnya, lihat [Mengkonfigurasi Kualitas Layanan untuk Pod](#) dalam dokumentasi Kubernetes.

Ketika Pod dijadwalkan di Fargate, reservasi vCPU dan memori dalam spesifikasi Pod menentukan berapa banyak CPU dan memori yang akan disediakan untuk Pod.

- Permintaan maksimum dari setiap kontainer Init digunakan untuk menentukan permintaan Init vCPU dan memori persyaratan.
- Permintaan untuk semua kontainer berjalan lama ditambahkan untuk menentukan lama berjalan permintaan vCPU dan memori persyaratan.
- Nilai yang lebih besar dari dua nilai sebelumnya dipilih untuk vCPU dan permintaan memori yang akan digunakan untuk Pod Anda.
- Fargate menambahkan 256 MB ke setiap reservasi memori Pod untuk komponen Kubernetes yang diperlukan (kubelet,, dan). kube-proxy containerd

Fargate membulatkan ke konfigurasi komputasi berikut yang paling cocok dengan jumlah permintaan vCPU dan memori untuk memastikan Pod selalu memiliki sumber daya yang mereka butuhkan untuk menjalankannya.

Jika Anda tidak menentukan kombinasi vCPU dan memori, maka kombinasi terkecil yang tersedia digunakan (0,25 vCPU dan memori 0,5 GB).

Tabel berikut menunjukkan kombinasi vCPU dan memori yang tersedia untuk Pod yang berjalan di Fargate.

Nilai vCPU	Nilai memori
0,25 vCPU	0,5 GB, 1 GB, 2 GB
0,5 vCPU	1 GB, 2 GB, 3 GB, 4 GB
1 vCPU	2 GB, 3 GB, 4 GB, 5 GB, 6 GB, 7 GB, 8 GB
2 vCPU	Antara 4 GB dan 16 GB dalam kenaikan 1-GB
4 vCPU	Antara 8 GB dan 30 GB dalam tambahan 1-GB
8 vCPU	Antara 16 GB dan 60 GB dengan peningkatan 4-GB
16 vCPU	Antara 32 GB dan 120 GB dengan peningkatan 8-GB

Memori tambahan yang disediakan untuk komponen Kubernetes dapat menyebabkan tugas Fargate dengan lebih banyak v CPUs dari yang diminta untuk disediakan. Misalnya, permintaan untuk 1 vCPU dan memori 8 GB akan memiliki 256 MB yang ditambahkan ke permintaan memorinya, dan akan menyediakan tugas Fargate dengan memori 2 v CPUs dan 9 GB, karena tidak ada tugas dengan 1 vCPU dan memori 9 GB yang tersedia.

Tidak ada korelasi antara ukuran Pod yang berjalan di Fargate dan ukuran node yang dilaporkan oleh Kubernetes dengan `kubectl get nodes`. Ukuran node yang dilaporkan seringkali lebih besar dari kapasitas Pod. Anda dapat memverifikasi kapasitas Pod dengan perintah berikut. Ganti *default* dengan namespace Pod Anda dan *pod-name* dengan nama Pod Anda.

```
kubectl describe pod --namespace default pod-name
```

Contoh output adalah sebagai berikut.

```
[...]
annotations:
  CapacityProvisioned: 0.25vCPU 0.5GB
[...]
```


CapacityProvisionedAnotasi tersebut mewakili kapasitas Pod yang diberlakukan dan menentukan biaya Pod Anda yang berjalan di Fargate. [Untuk informasi harga untuk konfigurasi komputasi, lihat Harga Fargate AWS](#).

Penyimpanan Fargate

Pod yang berjalan di Fargate secara otomatis memasang sistem file Amazon EFS, tanpa memerlukan langkah penginstalan driver manual. Anda tidak dapat menggunakan penyediaan volume persisten dinamis dengan node Fargate, tetapi Anda dapat menggunakan penyediaan statis. Untuk informasi selengkapnya, lihat [Driver Amazon EFS CSI](#) di GitHub.

Saat disediakan, setiap Pod yang berjalan di Fargate menerima penyimpanan sementara 20 GiB default. Jenis penyimpanan ini dihapus setelah Pod berhenti. Pod baru yang diluncurkan ke Fargate memiliki enkripsi volume penyimpanan sementara yang diaktifkan secara default. Penyimpanan Pod sementara dienkripsi dengan algoritma enkripsi AES-256 menggunakan kunci terkelola Fargate.

AWS

Note

Penyimpanan default yang dapat digunakan untuk Amazon EKS Pods yang berjalan di Fargate kurang dari 20 GiB. Ini karena beberapa ruang digunakan oleh modul Kubernetes kubelet dan modul Kubernetes lainnya yang dimuat di dalam Pod.

Anda dapat meningkatkan jumlah total penyimpanan sementara hingga maksimum 175 GiB. Untuk mengkonfigurasi ukuran dengan Kubernetes, tentukan permintaan ephemeral-storage sumber daya untuk setiap kontainer dalam sebuah Pod. Ketika Kubernetes menjadwalkan Pod, Kubernetes memastikan bahwa jumlah permintaan sumber daya untuk setiap Pod kurang dari kapasitas tugas Fargate. Untuk informasi selengkapnya, lihat [Manajemen Sumber Daya untuk Pod dan Container](#) di dokumentasi Kubernetes.

Amazon EKS Fargate menyediakan lebih banyak penyimpanan sementara daripada yang diminta untuk tujuan penggunaan sistem. Misalnya, permintaan 100 GiB akan menyediakan tugas Fargate dengan penyimpanan sementara 115 GiB.

Tetapkan tindakan untuk AWS acara patching OS Fargate

Amazon EKS secara berkala menambal OS untuk node AWS Fargate agar tetap aman. Sebagai bagian dari proses patching, kami mendaur ulang node untuk menginstal patch OS. Pembaruan

dicoba dengan cara yang menciptakan dampak paling kecil pada layanan Anda. Namun, jika Pod tidak berhasil diusir, ada kalanya Pod tersebut harus dihapus. Berikut ini adalah tindakan yang dapat Anda lakukan untuk meminimalkan potensi gangguan:

- Tetapkan anggaran gangguan Pod (PDBs) yang sesuai untuk mengontrol jumlah Pod yang turun secara bersamaan.
- Buat EventBridge aturan Amazon untuk menangani pengusuran yang gagal sebelum Pod dihapus.
- Mulai ulang pod Anda yang terpengaruh secara manual sebelum tanggal pengusuran diposting di notifikasi yang Anda terima.
- Buat konfigurasi notifikasi di Pemberitahuan AWS Pengguna.

Amazon EKS bekerja sama dengan komunitas Kubernetes untuk membuat perbaikan bug dan patch keamanan tersedia secepat mungkin. Semua Pod Fargate dimulai pada versi patch Kubernetes terbaru, yang tersedia dari Amazon EKS untuk versi Kubernetes dari kluster Anda. Jika Anda memiliki Pod dengan versi patch yang lebih lama, Amazon EKS mungkin mendaur ulangnya untuk memperbaruinya ke versi terbaru. Ini memastikan bahwa Pod Anda dilengkapi dengan pembaruan keamanan terbaru. Dengan begitu, jika ada masalah [Common Vulnerabilities and Exposures](#) (CVE) yang kritis, Anda tetap up to date untuk mengurangi risiko keamanan.

Saat OS AWS Fargate diperbarui, Amazon EKS akan mengirimkan pemberitahuan yang menyertakan sumber daya yang terpengaruh dan tanggal pengusuran pod yang akan datang. Jika tanggal pengusuran yang diberikan tidak nyaman, Anda memiliki opsi untuk me-restart pod yang terpengaruh secara manual sebelum tanggal pengusuran yang diposting di notifikasi. Pod apa pun yang dibuat sebelum waktu Anda menerima notifikasi akan dikenakan pengusuran. Lihat [Dokumentasi Kubernetes](#) untuk petunjuk lebih lanjut tentang cara me-restart pod Anda secara manual.

Untuk membatasi jumlah Pod yang turun pada satu waktu ketika Pod didaur ulang, Anda dapat mengatur anggaran gangguan Pod (). PDBs Anda dapat menggunakan PDBs untuk menentukan ketersediaan minimum berdasarkan persyaratan masing-masing aplikasi Anda sambil tetap mengizinkan pembaruan terjadi. Ketersediaan minimum PDB Anda harus kurang dari 100%. Untuk informasi selengkapnya, lihat [Menentukan Anggaran Gangguan untuk Aplikasi Anda](#) di Dokumentasi Kubernetes.

Amazon EKS menggunakan [Eviction API](#) untuk menguras Pod dengan aman sambil menghormati PDBs yang Anda tetapkan untuk aplikasi tersebut. Pod diusir oleh Availability Zone untuk

meminimalkan dampak. Jika penggusuran berhasil, Pod baru mendapatkan patch terbaru dan tidak diperlukan tindakan lebih lanjut.

Ketika penggusuran Pod gagal, Amazon EKS mengirimkan peristiwa ke akun Anda dengan rincian tentang Pod yang gagal penggusuran. Anda dapat menindaklanjuti pesan sebelum waktu penghentian yang dijadwalkan. Waktu spesifik bervariasi berdasarkan urgensi tambalan. Ketika tiba waktunya, Amazon EKS mencoba untuk mengusir Pod lagi. Namun, kali ini acara baru tidak dikirim jika penggusuran gagal. Jika penggusuran gagal lagi, Pod Anda yang ada akan dihapus secara berkala sehingga Pod baru dapat memiliki patch terbaru.

Berikut ini adalah contoh peristiwa yang diterima ketika penggusuran Pod gagal. Ini berisi rincian tentang cluster, nama Pod, namespace Pod, profil Fargate, dan waktu terminasi yang dijadwalkan.

```
{
  "version": "0",
  "id": "12345678-90ab-cdef-0123-4567890abcde",
  "detail-type": "EKS Fargate Pod Scheduled Termination",
  "source": "aws.eks",
  "account": "111122223333",
  "time": "2021-06-27T12:52:44Z",
  "region": "region-code",
  "resources": [
    "default/my-database-deployment"
  ],
  "detail": {
    "clusterName": "my-cluster",
    "fargateProfileName": "my-fargate-profile",
    "podName": "my-pod-name",
    "podNamespace": "default",
    "evictErrorMessage": "Cannot evict pod as it would violate the pod's disruption budget",
    "scheduledTerminationTime": "2021-06-30T12:52:44.832Z[UTC]"
  }
}
```

Selain itu, memiliki beberapa PDBs yang terkait dengan Pod dapat menyebabkan peristiwa kegagalan penggusuran. Acara ini mengembalikan pesan galat berikut.

```
"evictErrorMessage": "This pod has multiple PodDisruptionBudget, which the eviction subresource does not support",
```

Anda dapat membuat tindakan yang diinginkan berdasarkan acara ini. Misalnya, Anda dapat menyesuaikan anggaran gangguan Pod (PDB) untuk mengontrol cara Pod diusir. Lebih khusus lagi, misalkan Anda memulai dengan PDB yang menentukan persentase target Pod yang tersedia. Sebelum Pod dihentikan secara paksa selama upgrade, Anda dapat menyesuaikan PDB ke persentase Pod yang berbeda. Untuk menerima acara ini, Anda harus membuat EventBridge aturan Amazon di AWS akun dan AWS Wilayah tempat cluster tersebut berada. Aturan harus menggunakan pola Kustom berikut. Untuk informasi selengkapnya, lihat [Membuat EventBridge aturan Amazon yang bereaksi terhadap peristiwa](#) di Panduan EventBridge Pengguna Amazon.

```
{
  "source": ["aws.eks"],
  "detail-type": ["EKS Fargate Pod Scheduled Termination"]
}
```

Target yang sesuai dapat ditetapkan untuk acara untuk menangkapnya. Untuk daftar lengkap target yang tersedia, lihat [EventBridge target Amazon](#) di Panduan EventBridge Pengguna Amazon. Anda juga dapat membuat konfigurasi notifikasi di Pemberitahuan AWS Pengguna. Saat menggunakan AWS Management Console untuk membuat notifikasi, di bawah Aturan Acara, pilih Elastic Kubernetes Service (EKS) untuk nama AWS layanan dan EKS Fargate Pod Terjadwal Terminasi untuk tipe Event. Untuk informasi selengkapnya, lihat [Memulai Pemberitahuan AWS Pengguna](#) di Panduan AWS Pengguna Pemberitahuan Pengguna.

Lihat [FAQs: Pemberitahuan pengurusan Fargate Pod di AWS re:Post untuk pertanyaan umum tentang Pengurusan](#) Pod EKS.

Kumpulkan aplikasi AWS Fargate dan metrik penggunaan

Anda dapat mengumpulkan metrik sistem dan metrik CloudWatch penggunaan untuk AWS Fargate.

Metrik aplikasi

Untuk aplikasi yang berjalan di Amazon EKS dan AWS Fargate, Anda dapat menggunakan AWS Distro for OpenTelemetry (ADOT). ADOT memungkinkan Anda mengumpulkan metrik sistem dan mengirimkannya ke dasbor CloudWatch Container Insights. Untuk memulai ADOT untuk aplikasi yang berjalan di Fargate, [lihat CloudWatch Menggunakan Wawasan Kontainer AWS dengan Distro OpenTelemetry](#) untuk dokumentasi ADOT.

Metrik penggunaan

Anda dapat menggunakan metrik CloudWatch penggunaan untuk memberikan visibilitas ke dalam penggunaan sumber daya akun Anda. Gunakan metrik ini untuk memvisualisasikan penggunaan layanan Anda saat ini pada CloudWatch grafik dan dasbor.

AWS Metrik penggunaan Fargate sesuai dengan kuota layanan. AWS Anda dapat mengonfigurasi alarm yang memberi tahu Anda saat penggunaan mendekati kuota layanan. Untuk informasi lebih lanjut tentang kuota layanan Fargate, lihat [the section called “Kuota layanan”](#).

AWS Fargate menerbitkan metrik berikut di namespace. `AWS/Usage`

Metrik	Deskripsi
<code>ResourceCount</code>	Jumlah sumber daya tertentu yang berjalan di akun Anda. Sumber daya ditentukan oleh dimensi yang terkait dengan metrik.

Dimensi berikut digunakan untuk menyempurnakan metrik penggunaan yang diterbitkan oleh AWS Fargate.

Dimensi	Deskripsi
<code>Service</code>	Nama AWS layanan yang berisi sumber daya. Untuk metrik penggunaan AWS Fargate, nilai untuk dimensi ini adalah <code>Fargate</code>
<code>Type</code>	Jenis entitas yang dilaporkan. Saat ini, satu-satunya nilai yang valid untuk metrik penggunaan AWS Fargate adalah <code>Resource</code>
<code>Resource</code>	Jenis sumber daya yang sedang berjalan. Saat ini, AWS Fargate mengembalikan informasi tentang penggunaan Fargate On-Demand Anda. Nilai sumber daya untuk penggunaan Sesuai Permintaan Fargate adalah <code>OnDemand</code>.

Dimensi	Deskripsi
	[CATATAN] ===== Penggunaan Fargate On-Demand menggabungkan Pod Amazon EKS menggunakan Fargate, tugas Amazon ECS menggunakan tipe peluncuran Fargate dan tugas Amazon ECS menggunakan penyedia kapasitas. FARGATE =====
Class	Kelas sumber daya yang akan dilacak. Saat ini, AWS Fargate tidak menggunakan dimensi kelas.

Membuat CloudWatch alarm untuk memantau metrik penggunaan sumber daya Fargate

AWS Fargate menyediakan metrik CloudWatch penggunaan yang sesuai dengan kuota AWS layanan untuk penggunaan sumber daya Fargate On-Demand. Di konsol Service Quotas, Anda dapat memvisualisasikan penggunaan Anda pada grafik. Anda juga dapat mengonfigurasi alarm yang memberi tahu Anda ketika penggunaan mendekati kuota layanan. Untuk informasi selengkapnya, lihat [the section called “Kumpulkan metrik”](#).

Gunakan langkah-langkah berikut untuk membuat CloudWatch alarm berdasarkan metrik penggunaan sumber daya Fargate.

1. Buka konsol Service Quotas di <https://console.aws.amazon.com/servicequotas/>.
2. Di panel navigasi kiri, pilih AWS layanan.
3. Dari daftar AWS layanan, cari dan pilih AWS Fargate.
4. Dalam daftar Kuota Layanan, pilih kuota penggunaan Fargate yang ingin Anda buat alarm.
5. Di bagian CloudWatch alarm Amazon, pilih Buat.
6. Untuk Ambang batas Alarm, pilih persentase nilai kuota yang ingin Anda tetapkan sebagai nilai alarm.
7. Untuk Nama alarm, masukkan nama untuk alarm, lalu pilih Buat.

Mulai pencatatan AWS Fargate untuk klaster Anda

Amazon EKS di Fargate menawarkan router log bawaan berdasarkan Fluent Bit. Ini berarti Anda tidak secara eksplisit menjalankan container Fluent Bit sebagai sespan, tetapi Amazon menjalankannya untuk Anda. Yang harus Anda lakukan adalah mengkonfigurasi router log. Konfigurasi terjadi melalui dedicated ConfigMap yang harus memenuhi kriteria berikut:

- Bernama `aws-logging`
- Dibuat di namespace khusus yang disebut `aws-observability`
- Tidak dapat melebihi 5300 karakter.

Setelah Anda membuat ConfigMap, Amazon EKS di Fargate secara otomatis mendeteksi dan mengonfigurasi router log dengannya. Fargate menggunakan versi AWS untuk Fluent Bit, distribusi Fluent Bit yang sesuai dengan hulu yang dikelola oleh AWS. Untuk informasi lebih lanjut, lihat [AWS Fluent Bit on GitHub](#).

Router log memungkinkan Anda untuk menggunakan luasnya layanan di AWS untuk analisis log dan penyimpanan. Anda dapat melakukan streaming log dari Fargate langsung ke Amazon, CloudWatch Amazon OpenSearch Service. [Anda juga dapat melakukan streaming log ke tujuan seperti Amazon S3, Amazon Kinesis Data Streams, dan alat mitra melalui Amazon Data Firehose.](#)

- Profil Fargate yang sudah ada yang menentukan namespace Kubernetes yang sudah ada yang kamu gunakan untuk Pod Fargate. Untuk informasi selengkapnya, lihat [the section called “Langkah 3: Buat profil Fargate untuk cluster Anda”](#).
- Peran eksekusi Fargate Pod yang ada. Untuk informasi selengkapnya, lihat [the section called “Langkah 2: Buat peran eksekusi Fargate Pod”](#).

Konfigurasi router log

Important

Agar log berhasil dipublikasikan, harus ada akses jaringan dari VPC tempat klaster Anda berada ke tujuan log. Ini terutama menyangkut pengguna yang menyesuaikan aturan jalan keluar untuk VPC mereka. Untuk contoh menggunakan CloudWatch, lihat [Menggunakan CloudWatch Log dengan titik akhir VPC antarmuka](#) di Panduan Pengguna Amazon CloudWatch Logs.

Pada langkah-langkah berikut, ganti masing-masing *example value* dengan nilai Anda sendiri.

1. Buat namespace Kubernetes khusus bernama `aws-observability`

- a. Simpan konten berikut ini ke file bernama `aws-observability-namespace.yaml` pada komputer Anda. Nilai untuk `name` harus `aws-observability` dan `aws-observability: enabled` label diperlukan.

```
kind: Namespace
apiVersion: v1
metadata:
  name: aws-observability
  labels:
    aws-observability: enabled
```

- b. Buat namespace.

```
kubectl apply -f aws-observability-namespace.yaml
```

2. Buat ConfigMap dengan nilai Fluent Conf data untuk mengirimkan log kontainer ke tujuan. Fluent Conf adalah Fluent Bit, yang merupakan bahasa konfigurasi prosesor log yang cepat dan ringan yang digunakan untuk merutekan log kontainer ke tujuan log pilihan Anda. Untuk informasi lebih lanjut, lihat [File konfigurasi](#) dalam dokumentasi Fluent Bit.

 Important

Bagian utama yang termasuk dalam tipikal Fluent Conf adalah `Service`, `Input`, `Filter`, dan `Output`. Namun, router log Fargate hanya menerima:

- `Output` bagian `Filter` dan.
- Sebuah `Parser` bagian.

Jika Anda memberikan bagian lain, mereka akan ditolak.

Router log Fargate mengelola bagian `Service` dan `Input`. Ini memiliki `Input` bagian berikut, yang tidak dapat dimodifikasi dan tidak diperlukan di `AndaConfigMap`. Namun, Anda bisa mendapatkan wawasan darinya, seperti batas buffer memori dan tag yang diterapkan untuk log.

```
[INPUT]
  Name tail
```



```
Buffer_Max_Size 66KB
DB /var/log/flb_kube.db
Mem_Buf_Limit 45MB
Path /var/log/containers/*.log
Read_From_Head On
Refresh_Interval 10
Rotate_Wait 30
Skip_Long_Lines On
Tag kube.*
```

Saat membuat `ConfigMap`, pertimbangkan aturan berikut yang digunakan Fargate untuk memvalidasi bidang:

- `[FILTER]`, `[OUTPUT]`, dan seharusnya `[PARSER]` ditentukan di bawah setiap kunci yang sesuai. Contohnya, `[FILTER]` harus berada dalam `filters.conf`. Anda dapat memiliki satu atau lebih `[FILTER]` s di bawah `filters.conf`. `[PARSER]` Bagian `[OUTPUT]` dan juga harus berada di bawah kunci yang sesuai. Dengan menentukan beberapa `[OUTPUT]` bagian, Anda dapat merutekan log Anda ke tujuan yang berbeda secara bersamaan.
- Fargate memvalidasi kunci yang diperlukan untuk setiap bagian. Name dan match diperlukan untuk setiap `[FILTER]` dan `[OUTPUT]`. Name dan format diperlukan untuk setiap `[PARSER]`. Kunci peka terhadap huruf besar dan kecil.
- Variabel lingkungan seperti `${ENV_VAR}` tidak diizinkan di `ConfigMap`.
- Indentasi harus sama dengan arahan atau pasangan nilai kunci dalam setiap `filters.conf`, `output.conf`, dan `parsers.conf`. Pasangan nilai kunci yang harus menjorok lebih dari arahan.
- Fargate memvalidasi terhadap filter yang didukung berikut: `grep`, `parser`, `record_modifier`, `rewrite_tag` `throttlenest`, `modify` dan `kubernetes`
- Fargate memvalidasi output yang didukung berikut: `es`, `firehose`, `kinesis_firehose`, `cloudwatch`, `cloudwatch_logs`, dan `kinesis`.
- Setidaknya satu Output plugin yang didukung harus disediakan di `ConfigMap` untuk mengaktifkan logging. Filter dan Parser tidak diperlukan untuk mengaktifkan logging.

Anda juga dapat menjalankan Fluent Bit di Amazon EC2 menggunakan konfigurasi yang diinginkan untuk memecahkan masalah apa pun yang timbul dari validasi. Buat Anda `ConfigMap` menggunakan salah satu contoh berikut.

 Important

Amazon EKS Fargate logging tidak mendukung konfigurasi dinamis file. ConfigMap Setiap perubahan pada a hanya ConfigMap diterapkan pada Pod baru. Perubahan tidak diterapkan pada Pod yang ada.

Buat ConfigMap menggunakan contoh untuk tujuan log yang Anda inginkan.

 Note

Anda juga dapat menggunakan Amazon Kinesis Data Streams untuk tujuan log Anda. Jika Anda menggunakan Kinesis Data Streams, pastikan bahwa peran eksekusi pod telah `kinesis:PutRecords` diberikan izin. Untuk informasi selengkapnya, lihat Izin Amazon Kinesis [Data](#) Streams di Bit Lancar: Manual Resmi.

Example

CloudWatch

Anda memiliki dua opsi output saat menggunakan CloudWatch:

- [Plugin keluaran yang ditulis dalam C](#)
- [Plugin keluaran yang ditulis dalam Golang](#)

Contoh berikut menunjukkan cara menggunakan `cloudwatch_logs` plugin untuk mengirim log ke CloudWatch.

- a. Simpan konten berikut ini ke file bernama `aws-logging-cloudwatch-configmap.yaml`. Ganti *region-code* dengan AWS Wilayah tempat cluster Anda berada. Parameter di bawah [OUTPUT] diperlukan.

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
  namespace: aws-observability
data:
  flb_log_cw: "false" # Set to true to ship Fluent Bit process logs to
                    CloudWatch.
```

```

filters.conf: |
  [FILTER]
    Name parser
    Match *
    Key_name log
    Parser crio
  [FILTER]
    Name kubernetes
    Match kube.*
    Merge_Log On
    Keep_Log Off
    Buffer_Size 0
    Kube_Meta_Cache_TTL 300s
output.conf: |
  [OUTPUT]
    Name cloudwatch_logs
    Match kube.*
    region region-code
    log_group_name my-logs
    log_stream_prefix from-fluent-bit-
    log_retention_days 60
    auto_create_group true
parsers.conf: |
  [PARSER]
    Name crio
    Format Regex
    Regex ^(?<time>[^\ ]+) (?<stream>stdout|stderr) (?<logtag>P|F) (?
<log>.*)$
    Time_Key time
    Time_Format %Y-%m-%dT%H:%M:%S.%L%z

```

b. Menerapkan manifes ke kluster Anda.

```
kubectl apply -f aws-logging-cloudwatch-configmap.yaml
```

Amazon OpenSearch Service

Jika Anda ingin mengirim log ke Amazon OpenSearch Service, Anda dapat menggunakan [es](#) output, yang merupakan plugin yang ditulis dalam C. Contoh berikut menunjukkan cara menggunakan plugin untuk mengirim log ke OpenSearch.

- Simpan konten berikut ini ke file bernama `aws-logging-opensearch-configmap.yaml`. Ganti setiap *example value* dengan nilai-nilai Anda sendiri.

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
  namespace: aws-observability
data:
  output.conf: |
    [OUTPUT]
      Name es
      Match *
      Host search-example-gjxdcilagiprbqlqn42jsty66y.region-
code.es.amazonaws.com
      Port 443
      Index example
      Type example_type
      AWS_Auth On
      AWS_Region region-code
      tls On
```

b. Menerapkan manifes ke klaster Anda.

```
kubectl apply -f aws-logging-opensearch-configmap.yaml
```

Firehose

Anda memiliki dua opsi output saat mengirim log ke Firehose:

- [kinesis_firehose](#) — Plugin keluaran yang ditulis dalam C.
- [firehose](#) - Plugin keluaran yang ditulis dalam Golang.

Contoh berikut menunjukkan cara menggunakan `kinesis_firehose` plugin untuk mengirim log ke Firehose.

a. Simpan konten berikut ini ke file bernama `aws-logging-firehose-configmap.yaml`. Ganti `region-code` dengan AWS Wilayah tempat cluster Anda berada.

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
  namespace: aws-observability
data:
```

```
output.conf: |
  [OUTPUT]
  Name  kinesis_firehose
  Match *
  region region-code
  delivery_stream my-stream-firehose
```

b. Menerapkan manifes ke kluster Anda.

```
kubectl apply -f aws-logging-firehose-configmap.yaml
```

3. Siapkan izin untuk peran eksekusi Fargate Pod untuk mengirim log ke tujuan Anda.

a. Unduh kebijakan IAM untuk tujuan Anda ke komputer Anda.

Example

CloudWatch

Unduh kebijakan CloudWatch IAM ke komputer Anda. Anda juga dapat [melihat kebijakan](#) di GitHub.

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-eks-fluent-logging-examples/mainline/examples/fargate/cloudwatchlogs/permissions.json
```

Amazon OpenSearch Service

Unduh kebijakan OpenSearch IAM ke komputer Anda. Anda juga dapat [melihat kebijakan](#) di GitHub.

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-eks-fluent-logging-examples/mainline/examples/fargate/amazon-elasticsearch/permissions.json
```

Pastikan kontrol akses OpenSearch Dashboard dikonfigurasi dengan benar. OpenSearch Dashboard `all_access` role in harus memiliki peran eksekusi Fargate Pod dan peran IAM yang dipetakan. Pemetaan yang sama harus dilakukan untuk `security_manager` peran tersebut. Anda dapat menambahkan pemetaan sebelumnya dengan memilih Menu, kemudian, lalu `SecurityRoles`, dan kemudian memilih peran masing-masing. Untuk informasi selengkapnya, lihat [Bagaimana cara memecahkan masalah CloudWatch Log sehingga mengalir ke domain Amazon ES saya?](#) .

Firehose

Unduh kebijakan Firehose IAM ke komputer Anda. Anda juga dapat [melihat kebijakan](#) di GitHub.

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-eks-fluent-logging-examples/mainline/examples/fargate/kinesis-firehose/permissions.json
```

- b. Buat kebijakan IAM dari file kebijakan yang Anda unduh.

```
aws iam create-policy --policy-name eks-fargate-logging-policy --policy-document file://permissions.json
```

- c. Lampirkan kebijakan IAM ke peran eksekusi pod yang ditentukan untuk profil Fargate Anda dengan perintah berikut. Ganti **111122223333** dengan ID akun Anda. Ganti **AmazonEKSFargatePodExecutionRole** dengan peran eksekusi Pod Anda (untuk informasi selengkapnya, lihat [the section called “Langkah 2: Buat peran eksekusi Fargate Pod”](#)).

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::111122223333:policy/eks-fargate-logging-policy \
  --role-name AmazonEKSFargatePodExecutionRole
```

Dukungan filter Kubernetes

Filter Fluent Bit Kubernetes memungkinkan Anda menambahkan metadata Kubernetes ke file log Anda. Untuk informasi selengkapnya tentang filter, lihat [Kubernetes](#) di dokumentasi Fluent Bit. Anda dapat menerapkan filter menggunakan titik akhir server API.

```
filters.conf: |
  [FILTER]
    Name          kubernetes
    Match          kube.*
    Merge_Log      On
    Buffer_Size     0
    Kube_Meta_Cache_TTL 300s
```

⚠ Important

- Kube_URL, Kube_CA_File, Kube_Token_Command, dan Kube_Token_File merupakan parameter konfigurasi yang dimiliki layanan dan tidak boleh ditentukan. Amazon EKS Fargate mengisi nilai-nilai ini.
- Kube_Meta_Cache_TTL adalah waktu Fluent Bit menunggu hingga berkomunikasi dengan server API untuk metadata terbaru. Jika Kube_Meta_Cache_TTL tidak ditentukan, Amazon EKS Fargate menambahkan nilai default 30 menit untuk mengurangi beban pada server API.

Untuk mengirimkan log proses Fluent Bit ke akun Anda

Anda dapat secara opsional mengirimkan log proses Fluent Bit ke Amazon CloudWatch menggunakan yang berikut ini. ConfigMap Pengiriman log proses Fluent Bit ke CloudWatch membutuhkan biaya konsumsi dan penyimpanan log tambahan. Ganti *region-code* dengan AWS Wilayah tempat cluster Anda berada.

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
  namespace: aws-observability
  labels:
data:
  # Configuration files: server, input, filters and output
  # =====
  flb_log_cw: "true" # Ships Fluent Bit process logs to CloudWatch.

  output.conf: |
    [OUTPUT]
      Name cloudwatch
      Match kube.*
      region region-code
      log_group_name fluent-bit-cloudwatch
      log_stream_prefix from-fluent-bit-
      auto_create_group true
```

Log berada CloudWatch di AWS Wilayah yang sama dengan cluster. Nama grup log adalah *my-cluster-fluent-bit-logs* dan nama logstream Fluent Bit adalah. *fluent-bit-podname-pod-namespace*

Note

- Log proses dikirim hanya ketika proses Bit Lancar berhasil dimulai. Jika ada kegagalan saat memulai Fluent Bit, log proses terlewatkan. Anda hanya dapat mengirimkan log proses ke CloudWatch.
- Untuk men-debug log proses pengiriman ke akun Anda, Anda dapat menerapkan sebelumnya ConfigMap untuk mendapatkan log proses. Fluent Bit gagal memulai biasanya karena Anda ConfigMap tidak diurai atau diterima oleh Fluent Bit saat memulai.

Untuk menghentikan pengiriman log proses Fluent Bit

Pengiriman log proses Fluent Bit ke CloudWatch membutuhkan biaya konsumsi dan penyimpanan log tambahan. Untuk mengecualikan log proses dalam ConfigMap pengaturan yang ada, lakukan langkah-langkah berikut.

1. Temukan grup CloudWatch log yang dibuat secara otomatis untuk log proses Fluent Bit klaster Amazon EKS Anda setelah mengaktifkan logging Fargate. Ini mengikuti format *my-cluster-fluent-bit-logs*.
2. Hapus aliran CloudWatch log yang ada yang dibuat untuk setiap log proses Pod di grup CloudWatch log.
3. Edit ConfigMap dan atur `flb_log_cw`: `"false"`.
4. Mulai ulang semua Pod yang ada di klaster.

Aplikasi uji

1. Menerapkan contoh Pod.
 - a. Simpan konten berikut ini ke file bernama `sample-app.yaml` pada komputer Anda.

```
apiVersion: apps/v1
kind: Deployment
metadata:
```



```

name: sample-app
namespace: same-namespace-as-your-fargate-profile
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:latest
          ports:
            - name: http
              containerPort: 80

```

b. Menerapkan manifes ke klaster.

```
kubectl apply -f sample-app.yaml
```

2. Lihat log NGINX menggunakan tujuan yang Anda konfigurasi di ConfigMap

Pertimbangan ukuran

Kami menyarankan Anda merencanakan hingga 50 MB memori untuk router log. Jika Anda mengharapkan aplikasi Anda untuk menghasilkan catatan pada throughput yang sangat tinggi, maka Anda harus menyediakan memori hingga 100 MB.

Pemecahan Masalah

Untuk mengonfirmasi apakah fitur logging diaktifkan atau dinonaktifkan karena beberapa alasan, seperti tidak valid ConfigMap, dan mengapa fitur tersebut tidak valid, periksa peristiwa Pod Anda. `kubectl describe pod pod-name` Outputnya mungkin mencakup peristiwa Pod yang menjelaskan apakah logging diaktifkan atau tidak, seperti contoh keluaran berikut.

```

[...]
Annotations:          CapacityProvisioned: 0.25vCPU 0.5GB
                      Logging: LoggingDisabled: LOGGING_CONFIGMAP_NOT_FOUND

```

```
[...]
Events:
  Type    Reason          Age    From
  ----    -
  Warning LoggingDisabled <unknown> fargate-scheduler
           Disabled logging because aws-logging configmap was not found. configmap
           "aws-logging" not found
```

Peristiwa Pod bersifat sementara dengan periode waktu tergantung pada pengaturan. Anda juga dapat melihat anotasi Pod menggunakan `kubectl describe pod pod-name`. Dalam anotasi Pod, terdapat informasi tentang apakah fitur logging diaktifkan atau dinonaktifkan dan alasannya.

Pilih jenis instans EC2 node Amazon yang optimal

Amazon EC2 menyediakan berbagai pilihan jenis instance untuk node pekerja. Setiap jenis instans menawarkan kemampuan komputasi, memori, penyimpanan, dan jaringan yang berbeda. Setiap instance juga dikelompokkan dalam keluarga instance berdasarkan kemampuan ini. Untuk daftar, lihat [Jenis instans yang tersedia](#) di Panduan EC2 Pengguna Amazon. Amazon EKS merilis beberapa variasi Amazon EC2 AMLs untuk mengaktifkan dukungan. Untuk memastikan bahwa jenis instans yang Anda pilih kompatibel dengan Amazon EKS, pertimbangkan kriteria berikut.

- Semua Amazon EKS saat ini AMLs tidak mendukung mac keluarga.
- Amazon EKS lengan dan non-akselerasi AMLs tidak mendukung g3, g4, inf, dan p keluarga.
- Amazon EKS yang dipercepat AMLs tidak mendukung c, hpc, m, dan t keluarga.
- Untuk instans berbasis ARM, Amazon Linux 2023 (AL2023) hanya mendukung jenis instans yang menggunakan prosesor Graviton2 atau yang lebih baru. AL2023 tidak mendukung A1 instance.

Saat memilih di antara jenis instans yang didukung oleh Amazon EKS, pertimbangkan kemampuan masing-masing jenis berikut.

Jumlah instance dalam grup node

Secara umum, instance yang lebih sedikit dan lebih besar lebih baik, terutama jika Anda memiliki banyak Daemonset. Setiap instance memerlukan panggilan API ke server API, jadi semakin banyak instance yang Anda miliki, semakin banyak beban di server API.

Sistem operasi

Tinjau jenis instans yang didukung untuk [Linux](#), [Windows](#), dan [Bottlerocket](#). Sebelum membuat instance Windows, tinjau [Deploy node Windows di kluster EKS](#).

Arsitektur perangkat keras

Apakah Anda membutuhkan x86 atau Arm? Sebelum menerapkan instance Arm, tinjau [Arm Amazon Linux yang dioptimalkan oleh Amazon EKS](#). AMIs Apakah Anda memerlukan instance yang dibangun di atas Sistem Nitro ([Linux](#) atau [Windows](#)) atau yang memiliki kemampuan [Dipercepat](#)? Jika Anda membutuhkan kemampuan yang dipercepat, Anda hanya dapat menggunakan Linux dengan Amazon EKS.

Jumlah maksimum Pod

Karena setiap Pod diberi alamat IP sendiri, jumlah alamat IP yang didukung oleh tipe instance merupakan faktor dalam menentukan jumlah Pod yang dapat dijalankan pada instance. Untuk menentukan secara manual berapa banyak Pod yang didukung oleh tipe instance, lihat [the section called “Amazon EKS merekomendasikan Pod maksimum untuk setiap jenis EC2 instans Amazon”](#).

Note

Jika Anda menggunakan Amazon EKS yang dioptimalkan Amazon Linux 2 AMI v20220406 atau yang lebih baru, Anda dapat menggunakan jenis instans baru tanpa memutakhirkan ke AMI terbaru. Untuk ini AMIs, AMI secara otomatis menghitung max-pods nilai yang diperlukan jika tidak tercantum dalam [eni-max-podsfile.txt](#). Jenis instans yang saat ini dalam pratinjau mungkin tidak didukung oleh Amazon EKS secara default. Nilai max-pods untuk tipe tersebut masih perlu ditambahkan eni-max-pods.txt di AMI kami.

AWS Jenis instans [Sistem Nitro](#) secara opsional mendukung lebih banyak alamat IP daripada jenis instans Sistem non-Nitro. Namun, tidak semua alamat IP yang ditetapkan untuk sebuah instance tersedia untuk Pod. Untuk menetapkan jumlah alamat IP yang jauh lebih besar ke instans Anda, Anda harus memiliki versi 1.9.0 atau yang lebih baru dari add-on Amazon VPC CNI yang diinstal di cluster Anda dan dikonfigurasi dengan tepat. Untuk informasi selengkapnya, lihat [the section called “Meningkatkan alamat IP”](#). Untuk menetapkan jumlah alamat IP terbesar ke instans Anda, Anda harus memiliki versi 1.10.1 atau yang lebih baru dari add-on Amazon VPC CNI yang diinstal di cluster Anda dan menyebarkan cluster dengan keluarga. IPv6

Keluarga IP

Anda dapat menggunakan jenis instans apa pun yang didukung saat menggunakan IPv4 keluarga untuk klaster, yang memungkinkan klaster Anda menetapkan IPv4 alamat pribadi ke Pod dan Layanan Anda. Tetapi jika Anda ingin menggunakan IPv6 keluarga untuk cluster Anda, maka Anda harus menggunakan jenis instance [AWS Nitro System](#) atau tipe instance bare metal. Hanya IPv4 didukung untuk instance Windows. Cluster Anda harus menjalankan versi 1.10.1 atau yang lebih baru dari add-on Amazon VPC CNI. Untuk informasi selengkapnya tentang penggunaan IPv6, lihat [the section called “IPv6”](#).

Versi add-on Amazon VPC CNI yang Anda jalankan

[Versi terbaru plugin Amazon VPC CNI untuk Kubernetes mendukung jenis instance ini](#). Anda mungkin perlu memperbarui versi add-on Amazon VPC CNI untuk memanfaatkan jenis instans terbaru yang didukung. Untuk informasi selengkapnya, lihat [the section called “Amazon VPC CNI”](#). Versi terbaru mendukung fitur terbaru untuk digunakan dengan Amazon EKS. Versi sebelumnya tidak mendukung semua fitur. Anda dapat melihat fitur yang didukung oleh versi yang berbeda di [Changelog aktif](#). GitHub

AWS Wilayah tempat Anda membuat node

Tidak semua tipe instans tersedia di semua AWS Wilayah.

Apakah Anda menggunakan grup keamanan untuk Pod

Jika Anda menggunakan grup keamanan untuk Pod, hanya tipe instance tertentu yang didukung. Untuk informasi selengkapnya, lihat [the section called “Grup keamanan untuk Pod”](#).

Amazon EKS merekomendasikan Pod maksimum untuk setiap jenis EC2 instans Amazon

Karena setiap Pod diberi alamat IP sendiri, jumlah alamat IP yang didukung oleh tipe instance merupakan faktor dalam menentukan jumlah Pod yang dapat dijalankan pada instance. Amazon EKS menyediakan skrip yang dapat Anda unduh dan jalankan untuk menentukan jumlah maksimum Pod yang direkomendasikan Amazon EKS untuk dijalankan pada setiap jenis instans. Skrip menggunakan atribut perangkat keras dari setiap instance, dan opsi konfigurasi, untuk menentukan jumlah maksimum Pod. Anda dapat menggunakan nomor yang dikembalikan dalam langkah-langkah ini untuk mengaktifkan kemampuan seperti [menetapkan alamat IP ke Pod dari subnet yang berbeda dari instans](#) dan [secara signifikan meningkatkan jumlah alamat IP untuk instans Anda](#). Jika Anda

menggunakan grup node terkelola dengan beberapa tipe instance, gunakan nilai yang akan berfungsi untuk semua jenis instance.

1. Unduh skrip yang dapat Anda gunakan untuk menghitung jumlah maksimum Pod untuk setiap jenis instance.

```
curl -O https://raw.githubusercontent.com/awslabs/amazon-eks-ami/master/templates/al2/runtime/max-pods-calculator.sh
```

2. Tandai skrip sebagai dapat dieksekusi di komputer Anda.

```
chmod +x max-pods-calculator.sh
```

3. Jalankan skrip, ganti *m5.large* dengan jenis instans yang Anda rencanakan untuk diterapkan dan *1.9.0-eksbuild.1* dengan versi add-on Amazon VPC CNI Anda. Untuk menentukan versi add-on Anda, lihat prosedur pembaruan di [Menetapkan IPs ke Pod dengan CNI VPC Amazon](#).

```
./max-pods-calculator.sh --instance-type m5.large --cni-version 1.9.0-eksbuild.1
```

Contoh output adalah sebagai berikut.

```
29
```

Anda dapat menambahkan opsi berikut ke skrip untuk melihat Pod maksimum yang didukung saat menggunakan kemampuan opsional.

- `--cni-custom-networking-enabled`— Gunakan opsi ini ketika Anda ingin menetapkan alamat IP dari subnet yang berbeda dari instans Anda. Untuk informasi selengkapnya, lihat [the section called “Jaringan kustom”](#). Menambahkan opsi ini ke skrip sebelumnya dengan nilai contoh yang sama menghasilkan 20.
- `--cni-prefix-delegation-enabled`— Gunakan opsi ini ketika Anda ingin menetapkan lebih banyak alamat IP secara signifikan ke setiap elastic network interface. Kemampuan ini memerlukan instans Amazon Linux yang berjalan pada Sistem Nitro dan versi 1.9.0 atau yang lebih baru dari add-on Amazon VPC CNI. Untuk informasi selengkapnya, lihat [the section called “Meningkatkan alamat IP”](#). Menambahkan opsi ini ke skrip sebelumnya dengan nilai contoh yang sama menghasilkan 110.

Anda juga dapat menjalankan skrip dengan `--help` opsi untuk melihat semua opsi yang tersedia.

Note

Skrip kalkulator Pod max membatasi nilai pengembalian 110 berdasarkan [ambang batas skalabilitas Kubernetes dan pengaturan](#) yang direkomendasikan. Jika jenis instans Anda memiliki lebih dari 30 vCPUs, batas ini akan menjadi 250 angka berdasarkan pengujian tim skalabilitas Amazon EKS internal. Untuk informasi selengkapnya, lihat [plugin Amazon VPC CNI meningkatkan pod per node membatasi](#) posting blog.

Pertimbangan untuk Mode Otomatis EKS

Mode Otomatis EKS membatasi jumlah pod pada node ke yang lebih rendah dari:

- 110 polong tutup keras
- Hasil perhitungan pod maks yang dijelaskan di atas.

Buat node dengan gambar yang dioptimalkan sebelumnya

Anda dapat menerapkan node dengan Amazon EKS yang dioptimalkan [Amazon Machine Images](#) (AMIs) atau kustom Anda sendiri AMIs saat menggunakan grup node terkelola atau node yang dikelola sendiri. Jika Anda menjalankan node hybrid, lihat [the section called “Siapkan sistem operasi”](#). Untuk informasi tentang setiap jenis AMI Amazon EKS yang dioptimalkan, lihat salah satu topik berikut. Untuk petunjuk tentang cara membuat AMI kustom Anda sendiri, lihat [the section called “Build kustom”](#).

Dengan Amazon EKS Auto Mode, EKS mengelola EC2 instans termasuk memilih dan memperbarui AMI.

Topik

- [Panduan untuk fitur AMIs transisi EKS AL2 & AL2 -Dipercepat](#)
- [Buat node dengan Amazon Linux yang dioptimalkan AMIs](#)
- [Buat node dengan Bottlerocket yang dioptimalkan AMIs](#)
- [Buat node dengan Ubuntu Linux yang dioptimalkan AMIs](#)
- [Buat node dengan Windows yang dioptimalkan AMIs](#)

Panduan untuk fitur AMIs transisi EKS AL2 & AL2 -Dipercepat

AWS akan mengakhiri dukungan untuk EKS AL2 -dioptimalkan dan AL2 -dipercepat AMIs, efektif 26 November 2025. Meskipun Anda dapat terus menggunakan EKS AL2 AMIs setelah tanggal end-of-support (EOS) (26 November 2025), EKS tidak akan lagi merilis versi atau pembaruan Kubernetes baru AL2 AMIs, termasuk rilis minor, tambalan, dan perbaikan bug setelah tanggal ini. Kami merekomendasikan untuk meningkatkan ke Amazon Linux 2023 (AL2023) atau Bottlerocket AMIs

- AL2023 memungkinkan secure-by-default pendekatan dengan kebijakan keamanan yang telah dikonfigurasi sebelumnya, SELinux dalam mode permisif, mode IMDSv2 -only diaktifkan secara default, waktu boot yang dioptimalkan, dan manajemen paket yang ditingkatkan untuk meningkatkan keamanan dan kinerja, cocok untuk infrastruktur yang membutuhkan penyesuaian signifikan seperti akses tingkat OS langsung atau perubahan node yang ekstensif.
- Bottlerocket memungkinkan peningkatan keamanan, waktu boot yang lebih cepat, dan permukaan serangan yang lebih kecil untuk meningkatkan efisiensi dengan desain kontainer yang dibuat khusus dan dioptimalkan, sangat cocok untuk pendekatan asli kontainer dengan kustomisasi node minimal.

Atau, Anda bisa [the section called “Build kustom”](#) sampai tanggal EOS (26 November 2025), atau membuat AMI kustom dengan instans dasar Amazon Linux 2 hingga tanggal Amazon Linux 2 EOS (30 Juni 2026). Untuk informasi lebih lanjut, silakan kunjungi [AL2023 FAQs](#), [Bottlerocket FAQs](#) atau lihat [the section called “Tingkatkan ke AL2 023”](#) atau [the section called “Bottlerocket”](#) dokumentasi untuk panduan migrasi terperinci.

Migrasi dan dukungan FAQs

Bagaimana cara saya bermigrasi dari AMI saya AL2 ke AL2 023 AMI?

Sebaiknya buat dan implementasikan rencana migrasi yang mencakup pengujian beban kerja aplikasi menyeluruh dan prosedur rollback yang terdokumentasi, lalu ikuti step-by-step petunjuk dalam Upgrade [dari Amazon Linux 2 ke Amazon Linux 2023](#) dalam dokumentasi resmi EKS.

Dapatkah saya membuat AL2 AMI kustom melewati tanggal EKS end-of-support (EOS) untuk EKS yang dioptimalkan? AL2 AMIs

Meskipun kami merekomendasikan pindah ke EKS yang didukung dan dipublikasikan secara resmi yang dioptimalkan AMIs untuk AL2 023 atau Bottlerocket, Anda dapat membuat EKS kustom yang

diptimalkan AL2 dan -dipercepat AL2 hingga AMIs tanggal AL2 AMI EOS (26 November 2025). Atau, Anda dapat membuat AMI kustom dengan instans dasar Amazon Linux 2 hingga tanggal Amazon Linux 2 EOS (30 Juni 2026). Untuk step-by-step petunjuk untuk membuat AMI yang AL2 dioptimalkan dan dipercepat EKS kustom, lihat [Membuat AL2 AMI Amazon Linux kustom di dokumentasi](#) resmi EKS.

Apakah kebijakan dukungan versi EKS Kubernetes berlaku untuk distribusi Amazon Linux?

Tidak. Tanggal EOS untuk EKS yang AL2 dioptimalkan dan AL2 dipercepat AMIs tidak tergantung pada garis waktu dukungan standar dan diperpanjang untuk versi Kubernetes oleh EKS. Anda perlu bermigrasi ke AL2 023 atau Bottlerocket bahkan jika Anda menggunakan dukungan tambahan EKS.

Bagaimana pergeseran dari cgroupv1 ke cgroupv2 memengaruhi migrasi saya?

[Komunitas Kubernetes](#) memindahkan cgroupv1 dukungan (digunakan oleh AL2) ke mode pemeliharaan, yang berarti tidak ada fitur baru yang akan ditambahkan dan hanya keamanan kritis dan perbaikan bug utama yang akan disediakan. Untuk mengadopsi cgroupv2 di Kubernetes, Anda perlu memastikan kompatibilitas di seluruh komponen OS, kernel, runtime container, dan Kubernetes. Ini membutuhkan distribusi Linux yang memungkinkan secara cgroupv2 default, seperti AL2 023, Bottlerocket, Red Hat Enterprise Linux (RHEL) 9+, Ubuntu 22.04+, atau Debian 11+. Distribusi ini dikirimkan dengan versi kernel ≥ 5.8 , yang merupakan persyaratan minimum untuk cgroupv2 dukungan di Kubernetes. Untuk mempelajari lebih lanjut, lihat [Tentang cgroup v2](#).

Apa yang harus saya lakukan jika saya membutuhkan Neuron di AL2 AMI khusus saya?

Anda tidak dapat menjalankan aplikasi penuh bertenaga Neuron Anda secara native berbasis. AL2 AMIs Untuk memanfaatkan AWS Neuron pada AL2 AMI, Anda harus mengkontainerisasi aplikasi Anda menggunakan wadah yang didukung Neuron dengan distribusi AL2 non-Linux (misalnya, Ubuntu 22.04, Amazon Linux 2023, dll.) Dan kemudian menyebarkan kontainer tersebut pada AL2 AMI berbasis yang memiliki Neuron Driver () diinstal. `aws-neuronx-dkms`

Kompatibilitas dan versi

Versi Kubernetes yang didukung untuk AL2 AMIs

Kubernetes versi 1.32 adalah versi terakhir yang akan dirilis Amazon EKS (AL2 Amazon Linux 2). AMIs Untuk versi Kubernetes yang [didukung](#) hingga 1.32, EKS akan terus merilis AL2 AMIs (AL2_ARM_64, _x86_64) dan -accelerated (AL2_x86_64_GPU) hingga 26 November 2025. AL2 AMIs AL2 Setelah tanggal ini, EKS akan berhenti merilis AL2 -optimized dan AL2 -accelerated AMIs

untuk semua versi Kubernetes. Perhatikan bahwa tanggal EOS untuk EKS AL2 -optimized dan AL2 -accelerated AMIs tidak tergantung pada garis waktu dukungan standar dan diperpanjang untuk versi Kubernetes oleh EKS.

Perbandingan driver NVIDIA untuk AL2, AL2 023, dan Bottlerocket AMIs

Cabang Pengemudi	Amazon Linux 2 AMI	Amazon Linux 2023 AMI	Bottlerocket AMI	End-of-Life Tanggal
R535	Tidak Didukung	Tidak Didukung	Didukung	September 2027
R550	Didukung	Didukung	Tidak Didukung	April 2025
R560	Tidak Didukung	Didukung	Tidak Didukung	Maret 2025
R570	Tidak Didukung	Didukung	Segera hadir	Februari 2026

Untuk mempelajari lebih lanjut, lihat [Dokumentasi Rilis Nvidia](#).

Perbandingan versi NVIDIA CUDA untuk AL2, AL2 023, dan Bottlerocket AMIs

Versi CUDA	AL2 Support	AL2023 Support	Support Bottlerocket
10.1	Didukung	Tidak didukung	Tidak Didukung
11.8	Didukung	Didukung	Didukung
12.0	Tidak didukung	Didukung	Didukung
12,5	Tidak didukung	Didukung	Didukung

Untuk mempelajari lebih lanjut, lihat [Dokumentasi Rilis CUDA](#).

Driver yang didukung dan perbandingan versi kernel Linux untuk AL2, AL2 023, dan Bottlerocket AMIs

Komponen	AL2 Sumber AMI	AL2023 Sumber AMI	Sumber AMI Bottlerocket
Kompatibilitas OS Dasar	RHEL7/CentOS 7	Fedora/CentOS 9	N/A
Kit Alat CUDA	CUDA 11.x—12.x	CUDA 12,5+	CUDA 11.x (12,5 segera hadir)
Pengemudi GPU NVIDIA	R550	R565 (R570 segera hadir)	R535 (R570 Segera hadir)
AWS Pengemudi Neuron	2.19	2.19+	2.20
Kernel Linux	5.10	6.1, 6.12	5.15, 6.1 (6.12 segera hadir)

AWS Kompatibilitas neuron dengan AL2 AMIs

Mulai dari [rilis AWS Neuron 2.20](#), Neuron Runtime (`aws-neuronx-runtime-lib`) yang digunakan oleh EKS berbasis AL AMIs tidak lagi mendukung Amazon Linux 2 (). AL2 Neuron Driver (`aws-neuronx-dkms`) sekarang menjadi satu-satunya paket AWS Neuron yang mendukung Amazon Linux 2. Ini berarti Anda tidak dapat menjalankan aplikasi bertenaga Neuron Anda secara native pada AMI berbasis AL2. Untuk mengatur Neuron pada AL2 023 AMIs, lihat panduan [Pengaturan AWS Neuron](#).

Kompatibilitas Kubernetes dengan AL2 AMIs

Komunitas Kubernetes telah memindahkan `cgroupv1` dukungan (digunakan oleh AL2) ke mode pemeliharaan. Ini berarti tidak ada fitur baru yang akan ditambahkan, dan hanya keamanan kritis dan perbaikan bug utama yang akan disediakan. Fitur Kubernetes apa pun yang mengandalkan `cgroupv2`, seperti MemoryQoS dan isolasi sumber daya yang disempurnakan, tidak tersedia. AL2 Selanjutnya, Amazon EKS Kubernetes versi 1.32 adalah versi terakhir yang didukung. AL2 AMIs Untuk menjaga kompatibilitas dengan versi Kubernetes terbaru, sebaiknya migrasi ke AL2 023 atau Bottlerocket, yang diaktifkan secara default. `cgroupv2`

Kompatibilitas versi Linux dengan AL2 AMIs

Amazon Linux 2 (AL2) didukung AWS hingga tanggal end-of-support (EOS) pada 30 Juni 2026. Namun, seiring AL2 bertambahnya usia, dukungan dari komunitas Linux yang lebih luas untuk aplikasi dan fungsionalitas baru menjadi lebih terbatas. AL2 AMIs didasarkan pada [kernel Linux 5.10](#), sedangkan AL2 023 menggunakan [kernel Linux 6.10](#). Tidak seperti AL2 023, AL2 memiliki dukungan terbatas dari komunitas Linux yang lebih luas. Ini berarti banyak paket dan alat Linux hulu perlu di-backport untuk bekerja dengan AL2 versi kernel yang lebih lama, beberapa fitur Linux modern dan peningkatan keamanan tidak tersedia karena kernel yang lebih tua, banyak proyek open source telah usang atau dukungan terbatas untuk versi kernel yang lebih lama seperti 5.10.

Paket usang tidak termasuk dalam 023 AL2

Beberapa paket paling umum yang tidak disertakan atau yang diubah di AL2 023, termasuk:

- Beberapa [paket biner sumber di Amazon Linux 2](#) tidak lagi tersedia di Amazon Linux 2023
- Perubahan cara Amazon Linux mendukung versi paket yang berbeda (misalnya, [amazon-linux-extras sistem](#)) di AL2 023
- [Paket Ekstra untuk Enterprise Linux \(EPEL\)](#) tidak didukung di 023 AL2
- [Aplikasi 32-bit](#) tidak didukung di 023 AL2
- AL2023 tidak membuat paket [mate-desktop](#)

Untuk mempelajari lebih lanjut, lihat [Membandingkan AL2 dan AL2 023](#).

Perbandingan validasi FIPS di seluruh AL2, AL2 023, dan Bottlerocket

Amazon Linux 2 (AL2), Amazon Linux 2023 (AL2023), dan Bottlerocket memberikan dukungan untuk kepatuhan Federal Information Processing Standards (FIPS).

- AL2 disertifikasi di bawah FIPS 140-2 dan AL2 023 disertifikasi di bawah FIPS 140-3. Untuk mengaktifkan mode FIPS pada AL2 023, instal paket yang diperlukan di EC2 instans Amazon Anda dan ikuti langkah-langkah konfigurasi menggunakan instruksi di [Aktifkan Mode FIPS](#) di 023. AL2 Untuk mempelajari lebih lanjut, lihat [AL2023 FAQs](#).
- Bottlerocket menyediakan varian yang dibuat khusus untuk FIPS yang membatasi komponen kernel dan ruang pengguna untuk penggunaan modul kriptografi yang telah diserahkan ke Program Validasi Modul Kriptografi FIPS 140-3.

EKS AMI driver dan versi changelog

Untuk daftar lengkap semua komponen EKS AMI dan versinya, lihat [Catatan Rilis Amazon EKS AMI](#) di GitHub.

Buat node dengan Amazon Linux yang dioptimalkan AMIs

Amazon EKS yang dioptimalkan Amazon Linux AMIs dibangun di atas Amazon Linux 2 (AL2) dan Amazon Linux 2023 (AL2023). Mereka dikonfigurasi untuk berfungsi sebagai gambar dasar untuk node Amazon EKS. AMIs ini dikonfigurasi untuk bekerja dengan Amazon EKS dan mereka menyertakan komponen berikut:

- kubelet
- AWS Autentikator IAM
- containerd

Note

- Anda dapat melacak peristiwa keamanan atau privasi untuk Amazon Linux di [pusat keamanan Amazon Linux](#) dengan memilih tab untuk versi yang Anda inginkan. Anda juga dapat berlangganan umpan RSS yang berlaku. Kejadian keamanan dan privasi mencakup gambaran umum mengenai masalah, paket apa yang terpengaruh, dan cara memperbarui instans Anda untuk memperbaiki masalah tersebut.
- Sebelum menerapkan akselerasi atau Arm AMI, tinjau informasi di [Amazon EKS yang dioptimalkan dan dipercepat Amazon Linux AMIs](#) dan [the section called “Amazon EKS dioptimalkan Arm Amazon Linux AMIs”](#).
- EC2 P2Instans Amazon tidak didukung di Amazon EKS karena memerlukan NVIDIA driver versi 470 atau lebih lama.
- Setiap grup node terkelola yang baru dibuat dalam cluster pada versi 1.30 atau yang lebih baru akan secara otomatis default menggunakan AL2 023 sebagai sistem operasi node. Sebelumnya, grup node baru akan default ke AL2. Anda dapat terus menggunakannya AL2 dengan memilihnya sebagai tipe AMI saat membuat grup node baru.
- Amazon EKS tidak akan lagi menerbitkan Amazon Linux 2 (AL2) yang dioptimalkan EKS AMIs setelah 26 November 2025. Selain itu, versi Kubernetes 1.32 adalah versi terakhir

yang akan dirilis Amazon EKS. AL2 AMIs Dari versi dan 1.33 seterusnya, Amazon EKS akan terus merilis AL2 023 dan berbasis Bottlerocket. AMIs

Amazon EKS mengoptimalkan Amazon Linux yang dipercepat AMIs

Amazon EKS yang dioptimalkan Amazon Linux AMIs dipercepat dibangun di atas standar Amazon EKS yang dioptimalkan Amazon Linux AMIs. Mereka dikonfigurasi untuk berfungsi sebagai gambar opsional untuk node Amazon EKS untuk mendukung beban kerja berbasis GPU, [Inferentia](#), dan [Trainium](#).

Selain konfigurasi AMI standar Amazon EKS yang dioptimalkan, akselerasi AMIs meliputi yang berikut:

- Driver NVIDIA
- `nvidia-container-toolkit`
- AWS Pengemudi neuron

Untuk daftar komponen terbaru yang termasuk dalam akselerasi AMIs, lihat `amazon-eks-ami` [Rilis](#) di GitHub.

Note

- Pastikan untuk menentukan jenis instance yang berlaku di AWS CloudFormation template node Anda. Dengan menggunakan akselerasi Amazon EKS yang dioptimalkan AMIs, Anda menyetujui [Perjanjian Lisensi Pengguna Akhir Cloud \(EULA\) NVIDIA](#).
- Akselerasi Amazon EKS AMIs yang dioptimalkan sebelumnya disebut sebagai Amazon EKS yang dioptimalkan AMIs dengan dukungan GPU.
- Versi sebelumnya dari Amazon EKS yang dioptimalkan dipercepat AMIs menginstal `nvidia-docker` repositori. Repositori tidak lagi disertakan dalam Amazon EKS AMI versi v20200529 dan yang lebih baru.

Untuk detail tentang menjalankan beban kerja di Amazon EKS yang dioptimalkan Amazon Linux yang dipercepat AMIs, lihat [the section called “Jalankan Linux GPU AMIs”](#).

Amazon EKS dioptimalkan Arm Amazon Linux AMIs

Instans Arm menawarkan penghematan biaya yang signifikan untuk penskalaan dan aplikasi berbasis Arm, seperti server web, layanan mikro yang dikontainerisasi, armada caching, dan penyimpanan data terdistribusi. Saat menambahkan simpul Arm ke kluster Anda, kaji dulu pertimbangan-pertimbangan berikut ini.

- Jika kluster Anda diterapkan sebelum 17 Agustus 2020, Anda harus melakukan peningkatan satu kali dari manifes add-on cluster kritis. Ini agar Kubernetes dapat menarik gambar yang benar untuk setiap arsitektur perangkat keras yang digunakan di cluster Anda. Untuk informasi selengkapnya terkait cara memperbarui kluster add-on, lihat [the section called “Langkah 1: Bersiaplah untuk upgrade”](#). Jika Anda menerapkan cluster Anda pada atau setelah 17 Agustus 2020, maka plugin CoreDNS,, kube-proxy dan Amazon VPC CNI untuk add-on Kubernetes sudah memiliki kemampuan multi-arsitektur.
- Aplikasi yang di-deploy ke simpul Arm harus dikompilasi untuk Arm.
- Jika Anda memiliki DaemonSets yang di-deploy di kluster yang ada, atau Anda ingin menerapkannya ke kluster baru yang juga ingin Anda gunakan untuk menyebarkan node Arm, maka verifikasi bahwa Anda DaemonSet dapat berjalan di semua arsitektur perangkat keras di kluster Anda.
- Anda dapat menjalankan grup simpul Arm dan grup simpul x86 dalam kluster yang sama. Jika Anda melakukannya, pertimbangkan untuk menerapkan image container multi-arsitektur ke repositori kontainer seperti Amazon Elastic Container Registry dan kemudian menambahkan pemilih node ke manifes Anda sehingga Kubernetes mengetahui arsitektur perangkat keras apa yang dapat digunakan Pod. Untuk informasi selengkapnya, lihat [Mendorong gambar multi-arsitektur](#) di Panduan Pengguna Amazon ECR dan [Memperkenalkan gambar wadah multi-arsitektur untuk posting blog Amazon ECR](#).

Informasi selengkapnya

Untuk informasi selengkapnya tentang penggunaan Amazon EKS yang dioptimalkan Amazon Linux AMIs, lihat bagian berikut:

- Untuk menggunakan Amazon Linux dengan grup node terkelola, lihat [the section called “Grup simpul terkelola”](#).
- Untuk meluncurkan node Amazon Linux yang dikelola sendiri, lihat [the section called “Dapatkan yang terbaru IDs”](#).

- Untuk informasi versi, lihat [the section called “Dapatkan informasi versi”](#).
- Untuk mengambil Amazon Linux Amazon EKS terbaru IDs yang dioptimalkan AMIs, lihat [the section called “Dapatkan yang terbaru IDs”](#).
- Untuk skrip sumber terbuka yang digunakan untuk membangun Amazon EKS yang dioptimalkan AMIs, lihat. [the section called “Build kustom”](#)

Tingkatkan dari Amazon Linux 2 ke Amazon Linux 2023

Amazon EKS AMIs yang dioptimalkan tersedia dalam dua keluarga berdasarkan AL2 dan AL2 023. AL2023 adalah sistem operasi berbasis Linux baru yang dirancang untuk menyediakan lingkungan yang aman, stabil, dan berkinerja tinggi untuk aplikasi cloud Anda. Ini adalah generasi berikutnya dari Amazon Linux dari Amazon Web Services dan tersedia di semua versi Amazon EKS yang didukung.

AL2023 menawarkan beberapa perbaikan. AL2 Untuk perbandingan selengkapnya, lihat [Membandingkan AL2 dan Amazon Linux 2023](#) di Panduan Pengguna Amazon Linux 2023. Beberapa paket telah ditambahkan, ditingkatkan, dan dihapus dari AL2. Sangat disarankan untuk menguji aplikasi Anda dengan AL2 023 sebelum memutakhirkan. Untuk daftar semua perubahan paket di AL2 023, lihat [Perubahan paket di Amazon Linux 2023 di](#) Catatan Rilis Amazon Linux 2023.

Selain perubahan ini, Anda harus mengetahui hal-hal berikut:

- AL2023 memperkenalkan proses inisialisasi node baru nodeadm yang menggunakan skema konfigurasi YAMAL. Jika Anda menggunakan grup node yang dikelola sendiri atau AMI dengan template peluncuran, Anda sekarang harus menyediakan metadata kluster tambahan secara eksplisit saat membuat grup node baru. [Contoh](#) parameter minimum yang diperlukan adalah sebagai berikut, di mana `apiServerEndpoint`, `certificateAuthority`, dan layanan sekarang `cidr` diperlukan:

```
---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: my-cluster
    apiServerEndpoint: https://example.com
    certificateAuthority: Y2VydGlmawNhdGVBdXR0b3JpdHk=
    cidr: 10.100.0.0/16
```

Pada tahun AL2, metadata dari parameter ini ditemukan dari panggilan Amazon EKS `DescribeCluster` API. Dengan AL2 023, perilaku ini telah berubah karena panggilan API tambahan berisiko melambat selama peningkatan skala node besar. Perubahan ini tidak memengaruhi Anda jika Anda menggunakan grup node terkelola tanpa templat peluncuran atau jika Anda menggunakan Karpenter. Untuk informasi selengkapnya tentang `certificateAuthority` dan `cidr`, lihat [DescribeCluster](#) di Referensi API Amazon EKS.

- Untuk AL2 023, `nodeadm` juga mengubah format untuk menerapkan parameter ke `kubelet` untuk setiap node menggunakan [NodeConfigSpec](#). Pada tahun AL2, ini dilakukan dengan `--kubelet-extra-args` parameter. Ini biasanya digunakan untuk menambahkan label dan taints ke node. Contoh di bawah ini menunjukkan penerapan `maxPods` dan `--node-labels` ke node.

```
---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: test-cluster
    apiServerEndpoint: https://example.com
    certificateAuthority: Y2Vy dG l m a W N h d G V B d X R o b 3 J p d H k =
    cidr: 10.100.0.0/16
  kubelet:
    config:
      maxPods: 110
    flags:
      - --node-labels=karpenter.sh/capacity-type=on-demand,karpenter.sh/nodepool=test
```

- Versi Amazon VPC CNI 1.16.2 atau yang lebih besar diperlukan untuk 023. AL2
- AL2023 membutuhkan secara default IMDSv2. IMDSv2 memiliki beberapa manfaat yang membantu meningkatkan postur keamanan. Ini menggunakan metode otentikasi berorientasi sesi yang memerlukan pembuatan token rahasia dalam permintaan HTTP PUT sederhana untuk memulai sesi. Token sesi dapat berlaku di mana saja antara 1 detik dan 6 jam. Untuk informasi selengkapnya tentang cara transisi dari IMDSv1 ke IMDSv2, lihat [Transisi ke menggunakan Layanan Metadata Instans Versi 2](#) dan [Dapatkan manfaat lengkap IMDSv2 dan nonaktifkan IMDSv1 di seluruh infrastruktur Anda AWS](#). Jika Anda ingin menggunakannya IMDSv1, Anda masih dapat melakukannya dengan mengganti pengaturan secara manual menggunakan properti peluncuran opsi metadata instance.

Note

Untuk IMDSv2, jumlah hop default untuk grup node terkelola diatur ke 1. Ini berarti bahwa kontainer tidak akan memiliki akses ke kredensial node menggunakan IMDS. Jika Anda memerlukan akses kontainer ke kredensial node, Anda masih dapat melakukannya dengan mengganti secara manual `HttpPutResponseHopLimit` dalam [template EC2 peluncuran Amazon](#) kustom, meningkatkannya menjadi 2. Alternatifnya, Anda dapat menggunakan [Amazon EKS Pod Identity](#) untuk memberikan kredensialnya. IMDSv2

- AL2023 menampilkan generasi berikutnya dari hierarki kelompok kontrol terpadu (`cgroupv2`). `cgroupv2` digunakan untuk mengimplementasikan runtime kontainer, dan oleh `systemd`. Meskipun AL2 023 masih menyertakan kode yang dapat membuat sistem berjalan menggunakan `cgroupv1`, ini bukan konfigurasi yang direkomendasikan atau didukung. Konfigurasi ini akan sepenuhnya dihapus dalam rilis besar masa depan Amazon Linux.
- `eksctl` versi 0.176.0 atau lebih besar diperlukan `eksctl` untuk mendukung AL2 023.

Untuk grup node terkelola yang sudah ada sebelumnya, Anda dapat melakukan peningkatan di tempat atau peningkatan biru/hijau tergantung pada cara Anda menggunakan templat peluncuran:

- Jika Anda menggunakan AMI kustom dengan grup node terkelola, Anda dapat melakukan pemutakhiran di tempat dengan menukar ID AMI di template peluncuran. Anda harus memastikan bahwa aplikasi dan data pengguna Anda ditransfer ke AL2 023 terlebih dahulu sebelum melakukan strategi peningkatan ini.
- Jika Anda menggunakan grup node terkelola dengan templat peluncuran standar atau dengan templat peluncuran khusus yang tidak menentukan ID AMI, Anda diharuskan memutakhirkan menggunakan blue/green strategy. A blue/green pemutakhiran biasanya lebih kompleks dan melibatkan pembuatan grup node yang sama sekali baru di mana Anda akan menentukan AL2 023 sebagai tipe AMI. Grup node baru perlu dikonfigurasi dengan hati-hati untuk memastikan bahwa semua data kustom dari grup AL2 node kompatibel dengan OS baru. Setelah grup node baru diuji dan divalidasi dengan aplikasi Anda, Pod dapat dimigrasikan dari grup node lama ke grup node baru. Setelah migrasi selesai, Anda dapat menghapus grup node lama.

Jika Anda menggunakan Karpenter dan ingin menggunakan AL2 023, Anda harus memodifikasi `EC2NodeClass amiFamily` bidang dengan 023. AL2 Secara default, Drift diaktifkan di Karpenter.

Ini berarti bahwa setelah `amiFamily` bidang telah diubah, Karpenter akan secara otomatis memperbarui node pekerja Anda ke AMI terbaru jika tersedia.

Ambil informasi versi Amazon Linux AMI

Amazon EKS yang dioptimalkan Amazon Linux AMIs dioptimalkan diversi oleh versi Kubernetes dan tanggal rilis AMI dalam format berikut:

```
k8s_major_version.k8s_minor_version.k8s_patch_version-release_date
```

[Setiap rilis AMI mencakup berbagai versi kubelet, kernel Linux, dan containerd](#). Akselerasi AMIs juga mencakup berbagai versi driver NVIDIA. Anda dapat menemukan informasi versi ini di [Changelog](#) di GitHub

Ambil AMI Amazon Linux yang direkomendasikan IDs

Saat menerapkan node, Anda dapat menentukan ID untuk Amazon Machine Image (AMI) yang dioptimalkan Amazon EKS yang telah dibuat sebelumnya. Untuk mengambil ID AMI yang sesuai dengan konfigurasi yang Anda inginkan, kueri AWS Systems Manager Parameter Store API. Menggunakan API ini menghilangkan kebutuhan untuk secara manual mencari AMI Amazon EKS yang dioptimalkan IDs. Untuk informasi selengkapnya, lihat [GetParameter](#). [Prinsip IAM](#) yang Anda gunakan harus memiliki izin `ssm:GetParameter` IAM untuk mengambil metadata AMI Amazon EKS yang dioptimalkan.

Anda dapat mengambil ID gambar dari Amazon EKS terbaru yang dioptimalkan Amazon Linux AMI yang dioptimalkan dengan perintah berikut, yang menggunakan `image_id` sub-parameter. Buat modifikasi berikut pada perintah sesuai kebutuhan dan kemudian jalankan perintah yang dimodifikasi:

- Ganti `<kubernetes-version>` dengan [versi Amazon EKS](#) yang didukung.
- Ganti `ami-type` dengan salah satu opsi berikut. Untuk informasi tentang jenis EC2 instans Amazon, lihat [jenis EC2 instans Amazon](#).
 - Gunakan `amazon-linux-2023/x86_64/standard` untuk instans berbasis Amazon Linux 2023 (AL2023)x86.
 - Gunakan `amazon-linux-2023/arm64/standard` untuk instans AL2 023 ARM, seperti instance berbasis [AWS Graviton](#).
 - Gunakan `amazon-linux-2023/x86_64/nvidia` untuk instans x86 berbasis NVIDIA AL2 023 terbaru yang disetujui.

- Gunakan *amazon-linux-2023/arm64/nvidia* untuk instans arm64 berbasis NVIDIA AL2 023 terbaru yang disetujui.
- Gunakan *amazon-linux-2023/x86_64/neuron* untuk instans AL2 023 [AWS Neuron](#) terbaru.
- Gunakan *amazon-linux-2* untuk instans x86 berbasis Amazon Linux 2 (AL2).
- Gunakan *amazon-linux-2-arm64* untuk instance AL2 ARM, seperti instance berbasis [AWS Graviton](#).
- Gunakan *amazon-linux-2-gpu* untuk instans x86 berbasis [akselerasi AL2 perangkat keras](#) untuk beban kerja berbasis GPU NVIDIA, [Inferentia](#), dan [Trainium](#).
- Ganti <region-code> dengan [AWS Wilayah yang didukung Amazon EKS](#) yang Anda inginkan ID AMI.

```
aws ssm get-parameter --name /aws/service/eks/optimized-ami/<kubernetes-version>/<ami-type>/recommended/image_id \  
  --region <region-code> --query "Parameter.Value" --output text
```

Berikut adalah contoh perintah setelah penggantian placeholder dibuat.

```
aws ssm get-parameter --name /aws/service/eks/optimized-ami/1.31/amazon-  
linux-2023/x86_64/standard/recommended/image_id \  
  --region us-west-2 --query "Parameter.Value" --output text
```

Contoh output adalah sebagai berikut.

```
ami-1234567890abcdef0
```

Membangun AMI Amazon Linux khusus

Important

Amazon EKS tidak akan lagi menerbitkan Amazon Linux 2 (AL2) yang dioptimalkan EKS AMIs setelah 26 November 2025. Selain itu, versi Kubernetes 1.32 adalah versi terakhir yang akan dirilis Amazon EKS. AL2 AMIs Dari versi dan 1.33 seterusnya, Amazon EKS akan terus merilis AL2 023 dan berbasis Bottlerocket. AMIs

Amazon EKS dioptimalkan Amazon Linux (AL) AMIs dibangun di atas AL2 dan AL2 023, khusus untuk digunakan sebagai node di kluster Amazon EKS. Amazon EKS menyediakan skrip build open-source di repositori [Amazon EKS AMI Build Specification](#) yang dapat Anda gunakan dengan cara berikut: * Lihat konfigurasi untuk `kubelet`, runtime, dan IAM Authenticator for Kubernetes. AWS * Bangun AMI berbasis AL Anda sendiri dari awal.

Repositori ini berisi [skrip bootstrap khusus dan skrip nodeadm](#) yang berjalan pada saat boot. Skrip ini mengonfigurasi data sertifikat instans Anda, titik akhir bidang kontrol, nama cluster, dan lainnya. Skrip dianggap sebagai sumber kebenaran untuk build AMI yang dioptimalkan Amazon EKS, sehingga Anda dapat mengikuti GitHub repositori untuk memantau perubahan pada kami. AMIs

Prasyarat

- [Instal AWS CLI](#)
- [Instal HashiCorp Packer v1.9.4 +](#)
- [Instal GNU Make](#)

Mulai cepat

Bagian ini menunjukkan kepada Anda perintah untuk membuat AMI kustom di AWS akun Anda. Untuk mempelajari lebih lanjut tentang konfigurasi yang tersedia untuk menyesuaikan AMI Anda, lihat variabel templat di halaman [Amazon Linux 2023](#).

Langkah 1. Siapkan lingkungan Anda

Kloning atau fork repositori Amazon EKS AMI resmi. Misalnya:

```
git clone https://github.com/aws-labs/amazon-eks-ami.git
cd amazon-eks-ami
```

Verifikasi bahwa Packer diinstal:

```
packer --version
```

Langkah 2. Buat AMI kustom

Berikut ini adalah contoh perintah untuk berbagai kustom AMIs.

Dasar NVIDIA AL2 AMI:

```
make k8s=1.31 os_distro=al2 \  
  enable_accelerator=nvidia \  
  nvidia_driver_major_version=560 \  
  enable_efa=true
```

Dasar NVIDIA AL2 023 AMI:

```
make k8s=1.31 os_distro=al2023 \  
  enable_accelerator=nvidia \  
  nvidia_driver_major_version=560 \  
  enable_efa=true
```

Neuron 023 AL2 AMI yang sesuai dengan STIG:

```
make k8s=1.31 os_distro=al2023 \  
  enable_accelerator=neuron \  
  enable_fips=true \  
  source_ami_id=ami-0abcd1234efgh5678 \  
  kms_key_id=alias/aws-stig
```

Setelah Anda menjalankan perintah ini, Packer akan melakukan hal berikut: * Luncurkan EC2 instance Amazon sementara. * Instal komponen, driver, dan konfigurasi Kubernetes. * Buat AMI di AWS akun Anda.

Langkah 3. Lihat nilai default

Untuk melihat nilai default dan opsi tambahan, jalankan perintah berikut:

```
make help
```

Buat node dengan Bottlerocket yang dioptimalkan AMIs

[Bottlerocket](#) adalah distribusi Linux open source yang disponsori dan didukung oleh AWS. Bottlerocket dibuat khusus untuk menampung beban kerja kontainer. Dengan Bottlerocket, Anda dapat meningkatkan ketersediaan penerapan kontainer dan mengurangi biaya operasional dengan mengotomatiskan pembaruan infrastruktur kontainer Anda. Bottlerocket hanya mencakup perangkat lunak penting untuk menjalankan kontainer, yang meningkatkan penggunaan sumber daya, mengurangi ancaman keamanan, dan menurunkan overhead manajemen. Bottlerocket AMI

containerd termasuk,kubelet, dan IAM Authenticator. AWS [Selain grup node dikelola dan node yang dikelola sendiri, Bottlerocket juga didukung oleh Karpenter.](#)

Keuntungan

Menggunakan Bottlerocket dengan cluster Amazon EKS Anda memiliki keuntungan sebagai berikut:

- Uptime yang lebih tinggi dengan biaya operasional yang lebih rendah dan kompleksitas manajemen yang lebih rendah — Bottlerocket memiliki jejak sumber daya yang lebih kecil, waktu boot yang lebih pendek, dan kurang rentan terhadap ancaman keamanan daripada distribusi Linux lainnya. Jejak Bottlerocket yang lebih kecil membantu mengurangi biaya dengan menggunakan lebih sedikit sumber daya penyimpanan, komputasi, dan jaringan.
- Peningkatan keamanan dari pembaruan OS otomatis - Pembaruan untuk Bottlerocket diterapkan sebagai satu unit yang dapat diputar kembali, jika perlu. Ini menghilangkan risiko pembaruan yang rusak atau gagal yang dapat membuat sistem dalam keadaan tidak dapat digunakan. Dengan Bottlerocket, pembaruan keamanan dapat diterapkan secara otomatis segera setelah tersedia dengan cara yang minimal mengganggu dan dibatalkan jika terjadi kegagalan.
- Dukungan premium - AWS disediakan build Bottlerocket di EC2 Amazon tercakup dalam paket AWS Support yang sama yang juga mencakup AWS layanan seperti Amazon, Amazon EKS EC2, dan Amazon ECR.

Pertimbangan

Pertimbangkan hal berikut saat menggunakan Bottlerocket untuk tipe AMI Anda:

- Bottlerocket mendukung instans EC2 x86_64 Amazon dengan dan prosesor. a1m64 AMI Bottlerocket tidak direkomendasikan untuk digunakan dengan instans EC2 Amazon dengan chip Inferentia.
- Gambar bottlerocket tidak menyertakan server SSH atau shell. Anda dapat menggunakan metode out-of-band akses untuk memungkinkan SSH. Pendekatan ini memungkinkan penampung admin dan meneruskan beberapa langkah konfigurasi bootstrap dengan data pengguna. Untuk informasi lebih lanjut, lihat bagian berikut di [Bottlerocket](#) OS di: GitHub
 - [Eksplorasi](#)
 - [Kontainer admin](#)
 - [Pengaturan Kubernetes](#)
- Bottlerocket menggunakan berbagai jenis wadah:

- Secara default, [wadah kontrol](#) diaktifkan. Container ini menjalankan [agen AWS Systems Manager](#) yang dapat Anda gunakan untuk menjalankan perintah atau memulai sesi shell di instance Amazon EC2 Bottlerocket. Untuk informasi selengkapnya, lihat [Menyiapkan Session Manager](#) di Panduan Pengguna AWS Systems Manager.
- Jika kunci SSH diberikan saat membuat grup node, wadah admin diaktifkan. Sebaiknya gunakan wadah admin hanya untuk skenario pengembangan dan pengujian. Kami tidak menyarankan menggunakannya untuk lingkungan produksi. Untuk informasi selengkapnya, lihat [Kontainer admin](#) di GitHub.

Informasi selengkapnya

Untuk informasi selengkapnya tentang penggunaan Bottlerocket yang dioptimalkan Amazon EKS AMIs, lihat bagian berikut:

- [Untuk detail tentang Bottlerocket, lihat Dokumentasi Bottlerocket.](#)
- Untuk sumber informasi versi, lihat [the section called “Dapatkan informasi versi”](#).
- Untuk menggunakan Bottlerocket dengan grup node terkelola, lihat. [the section called “Grup simpul terkelola”](#)
- Untuk meluncurkan node Bottlerocket yang dikelola sendiri, lihat. [the section called “Bottlerocket”](#)
- Untuk mengambil AMIs Bottlerocket Amazon EKS terbaru IDs yang dioptimalkan, lihat. [the section called “Dapatkan yang terbaru IDs”](#)
- Untuk detail tentang dukungan kepatuhan, lihat [the section called “Dukungan kepatuhan”](#).

Ambil informasi versi AMI Bottlerocket

[Setiap rilis AMI Bottlerocket mencakup berbagai versi kubelet, kernel Bottlerocket, dan containerd.](#)

Varian AMI yang dipercepat juga mencakup berbagai versi driver NVIDIA. Anda dapat menemukan informasi versi ini di topik [OS Dokumentasi](#) Bottlerocket. Dari halaman ini, navigasikan ke sub-topik Informasi Versi yang berlaku.

Dokumentasi Bottlerocket terkadang tertinggal dari versi yang tersedia di. GitHub Anda dapat menemukan daftar perubahan untuk versi terbaru dalam [rilis](#) di GitHub.

Ambil Bottlerocket AMI yang direkomendasikan IDs

Saat menerapkan node, Anda dapat menentukan ID untuk Amazon Machine Image (AMI) yang dioptimalkan Amazon EKS yang telah dibuat sebelumnya. Untuk mengambil ID AMI yang sesuai

dengan konfigurasi yang Anda inginkan, kueri AWS Systems Manager Parameter Store API. Menggunakan API ini menghilangkan kebutuhan untuk secara manual mencari AMI Amazon EKS yang dioptimalkan IDs. Untuk informasi selengkapnya, lihat [GetParameter](#). [Prinsip IAM](#) yang Anda gunakan harus memiliki izin `ssm:GetParameter` IAM untuk mengambil metadata AMI Amazon EKS yang dioptimalkan.

Anda dapat mengambil ID gambar dari AMI Bottlerocket Amazon EKS terbaru yang dioptimalkan dengan perintah AWS CLI berikut, yang menggunakan sub-parameter. `image_id` Buat modifikasi berikut pada perintah sesuai kebutuhan dan kemudian jalankan perintah yang dimodifikasi:

- Ganti *kubernetes-version* dengan [versi Amazon EKS](#) yang didukung.
- Ganti *-flavor* dengan salah satu opsi berikut.
 - Hapus *-flavor* untuk varian tanpa GPU.
 - Gunakan *-nvidia* untuk varian berkemampuan GPU.
 - Gunakan *-fips* untuk varian yang mendukung FIPS.
- Ganti *architecture* dengan salah satu opsi berikut.
 - Gunakan *x86_64* untuk instance x86 berbasis.
 - Gunakan *arm64* untuk instance ARM.
- Ganti *region-code* dengan [AWS Wilayah yang didukung Amazon EKS](#) yang Anda inginkan ID AMI.

```
aws ssm get-parameter --name /aws/service/bottlerocket/aws-k8s-kubernetes-version-flavor/architecture/latest/image_id \  
    --region region-code --query "Parameter.Value" --output text
```

Berikut adalah contoh perintah setelah penggantian placeholder dibuat.

```
aws ssm get-parameter --name /aws/service/bottlerocket/aws-k8s-1.31/x86_64/latest/  
image_id \  
    --region us-west-2 --query "Parameter.Value" --output text
```

Contoh output adalah sebagai berikut.

```
ami-1234567890abcdef0
```


Memenuhi persyaratan kepatuhan dengan Bottlerocket

Bottlerocket mematuhi rekomendasi yang ditentukan oleh berbagai organisasi:

- Ada [Tolok Ukur CIS](#) yang didefinisikan untuk Bottlerocket. Dalam konfigurasi default, gambar Bottlerocket memiliki sebagian besar kontrol yang diperlukan oleh profil konfigurasi CIS Level 1. Anda dapat menerapkan kontrol yang diperlukan untuk profil konfigurasi CIS Level 2. Untuk informasi lebih lanjut, lihat [Memvalidasi Amazon EKS Bottlerocket AMI yang dioptimalkan terhadap Tolok Ukur CIS di blog](#). AWS
- Set fitur yang dioptimalkan dan permukaan serangan yang dikurangi berarti bahwa instance Bottlerocket memerlukan lebih sedikit konfigurasi untuk memenuhi persyaratan PCI DSS. [Tolok Ukur CIS untuk Bottlerocket](#) adalah sumber yang sangat baik untuk panduan pengerasan, dan mendukung persyaratan Anda untuk standar konfigurasi yang aman di bawah persyaratan PCI DSS 2.2. Anda juga dapat memanfaatkan [Fluent Bit](#) untuk mendukung persyaratan Anda untuk pencatatan audit tingkat sistem operasi berdasarkan persyaratan PCI DSS 10.2. AWS menerbitkan instans Bottlerocket baru (ditambah) secara berkala untuk membantu Anda memenuhi persyaratan PCI DSS 6.2 (untuk v3.2.1) dan persyaratan 6.3.3 (untuk v4.0).
- Bottlerocket adalah fitur yang memenuhi syarat HIPAA yang diizinkan untuk digunakan dengan beban kerja yang diatur untuk Amazon dan Amazon EKS. EC2 Untuk informasi selengkapnya, lihat [Referensi Layanan yang Memenuhi Syarat HIPAA](#).
- Bottlerocket tersedia yang AMIs telah dikonfigurasi sebelumnya untuk menggunakan modul kriptografi tervalidasi FIPS 140-3. Ini termasuk Modul Kriptografi Amazon Linux 2023 Kernel Crypto API dan Modul Kriptografi AWS-LC. Lihat informasi yang lebih lengkap di [the section called "Bottlerocket FIPS AMIs"](#).

Buat FIPS node pekerja Anda siap dengan Bottlerocket FIPS AMIs

Federal Information Processing Standard (FIPS) Publikasi 140-3 adalah standar pemerintah Amerika Serikat dan Kanada yang menetapkan persyaratan keamanan untuk modul kriptografi yang melindungi informasi sensitif. Bottlerocket membuatnya lebih mudah untuk mematuhi FIPS dengan menawarkan AMIs dengan kernel FIPS.

Ini telah AMIs dikonfigurasi sebelumnya untuk menggunakan modul kriptografi tervalidasi FIPS 140-3. Ini termasuk Modul Kriptografi Amazon Linux 2023 Kernel Crypto API dan Modul Kriptografi AWS-LC.

Menggunakan Bottlerocket FIPS AMIs membuat node pekerja Anda “FIPS siap” tetapi tidak secara otomatis “FIPS-compliant”. Untuk informasi lebih lanjut, lihat [Federal Information Processing Standard \(FIPS\) 140-3](#).

Pertimbangan

- Jika klaster Anda menggunakan subnet terisolasi, titik akhir Amazon ECR FIPS mungkin tidak dapat diakses. Hal ini dapat menyebabkan node bootstrap gagal. Pastikan konfigurasi jaringan Anda memungkinkan akses ke titik akhir FIPS yang diperlukan. Untuk informasi selengkapnya, lihat [Mengakses sumber daya melalui titik akhir VPC sumber daya di Panduan](#). AWS PrivateLink
- Jika klaster Anda menggunakan subnet dengan [PrivateLink](#), penarikan gambar akan gagal karena titik akhir Amazon ECR FIPS tidak tersedia. PrivateLink

Buat grup node terkelola dengan AMI FIPS Bottlerocket

Bottlerocket FIPS AMI hadir dalam dua varian untuk mendukung beban kerja Anda:

- BOTTLEROCKET_x86_64_FIPS
- BOTTLEROCKET_ARM_64_FIPS

Untuk membuat grup node terkelola dengan AMI FIPS Bottlerocket, pilih tipe AMI yang berlaku selama proses pembuatan. Untuk informasi selengkapnya, lihat [the section called “Buat”](#).

Untuk informasi selengkapnya tentang memilih varian berkemampuan FIPS, lihat [the section called “Dapatkan yang terbaru IDs”](#)

Nonaktifkan titik akhir FIPS untuk Wilayah yang tidak didukung AWS

Bottlerocket FIPS AMIs didukung langsung di Amerika Serikat, termasuk Wilayah (AS). AWS GovCloud Untuk AWS Wilayah di mana AMIs tersedia tetapi tidak didukung secara langsung, Anda masih dapat menggunakan AMIs dengan membuat grup node terkelola dengan templat peluncuran.

Bottlerocket FIPS AMI bergantung pada titik akhir Amazon ECR FIPS selama bootstrap, yang umumnya tidak tersedia di luar Amerika Serikat. Untuk menggunakan AMI untuk kernel FIPS di AWS Wilayah yang tidak memiliki titik akhir Amazon ECR FIPS, lakukan langkah-langkah berikut untuk menonaktifkan titik akhir FIPS:

1. Buat file konfigurasi baru dengan konten berikut atau sertakan konten ke dalam file konfigurasi yang ada.

```
[default]
use_fips_endpoint=false
```

1. Encode konten file sebagai format Base64.
2. Di template peluncuran Anda UserData, tambahkan string yang dikodekan berikut menggunakan format TOLL:

```
[settings.aws]
config = "<your-base64-encoded-string>"
```

Untuk pengaturan lainnya, lihat [Deskripsi](#) pengaturan Bottlerocket aktif. GitHub

Berikut adalah contoh UserData dalam template peluncuran:

```
[settings]
motd = "Hello from eksctl!"
[settings.aws]
config = "W2RlZmF1bHRdCnVzZV9maXBzX2VuZHBvaW50PWZhbnHNlCg==" # Base64-encoded string.
[settings.kubernetes]
api-server = "<api-server-endpoint>"
cluster-certificate = "<cluster-certificate-authority>"
cluster-name = "<cluster-name>"
...<other-settings>
```

Untuk informasi selengkapnya tentang membuat template peluncuran dengan data pengguna, lihat [the section called “Templat peluncuran”](#).

Buat node dengan Ubuntu Linux yang dioptimalkan AMIs

Canonical telah bermitra dengan Amazon EKS untuk membuat node AMIs yang dapat Anda gunakan di cluster Anda.

[Canonical memberikan image](#) built-for-purpose Kubernetes Node OS. Gambar Ubuntu yang diminimalkan ini dioptimalkan untuk Amazon EKS dan menyertakan AWS kernel khusus yang dikembangkan bersama. AWS Untuk informasi selengkapnya, lihat [Ubuntu di Amazon Elastic Kubernetes Service](#) (EKS) dan [the section called “Ubuntu Linux”](#) Untuk informasi tentang dukungan, lihat bagian [Perangkat lunak pihak ketiga](#) dari Dukungan AWS Premium FAQs.

Buat node dengan Windows yang dioptimalkan AMIs

Windows Amazon EKS AMIs yang dioptimalkan dibangun di atas Windows Server 2019 dan Windows Server 2022. Mereka dikonfigurasi untuk berfungsi sebagai gambar dasar untuk node Amazon EKS. Secara default, AMIs termasuk komponen-komponen berikut:

- [kubelet](#)
- [kube-proxy](#)
- [AWS Authenticator IAM untuk Kubernetes](#)
- [csi-proxy](#)
- [containerd](#)

Note

Anda dapat melacak peristiwa keamanan atau privasi untuk Windows Server dengan [Microsoft security update guide](#).

Amazon EKS menawarkan AMIs yang dioptimalkan untuk wadah Windows dalam varian berikut:

- AMI Inti Windows Server 2019 yang dioptimalkan Amazon EKS
- Amazon EKS yang dioptimalkan Windows Server 2019 AMI Penuh
- Server Windows 2022 AMI Inti yang dioptimalkan Amazon EKS
- Amazon EKS yang dioptimalkan Windows Server 2022 AMI Penuh

Important

- Windows Server 20H2 Core AMI yang dioptimalkan Amazon EKS tidak digunakan lagi. Tidak ada versi baru dari AMI ini yang akan dirilis.
- Untuk memastikan bahwa Anda memiliki pembaruan keamanan terbaru secara default, Amazon EKS mempertahankan Windows AMIs yang dioptimalkan selama 4 bulan terakhir. Setiap AMI baru akan tersedia selama 4 bulan sejak rilis awal. Setelah periode ini, yang lebih tua AMIs dibuat pribadi dan tidak lagi dapat diakses. Kami mendorong penggunaan yang terbaru AMIs untuk menghindari kerentanan keamanan dan kehilangan akses

ke AMIs yang lebih tua yang telah mencapai akhir masa pakai yang didukung mereka. Meskipun kami tidak dapat menjamin bahwa kami dapat menyediakan akses ke AMIs yang telah dibuat pribadi, Anda dapat meminta akses dengan mengajukan tiket ke AWS Support.

Kalender rilis

Tabel berikut mencantumkan rilis dan tanggal akhir dukungan untuk versi Windows di Amazon EKS. Jika tanggal akhir kosong, itu karena versi masih didukung.

Versi Windows	Rilisan Amazon EKS	Akhir dukungan Amazon EKS
Windows Server 2022 Inti	10/17/2022	
Windows Server 2022 Lengkap	10/17/2022	
Windows Server 20H2 Inti	8/12/2021	8/9/2022
Windows Server 2004 Core	8/19/2020	12/14/2021
Windows Server 2019 Core	10/7/2019	
Windows Server 2019 Full	10/7/2019	
Windows Server 1909 Core	10/7/2019	12/8/2020

Parameter konfigurasi skrip bootstrap

Saat Anda membuat node Windows, ada skrip pada node yang memungkinkan untuk mengkonfigurasi parameter yang berbeda. Bergantung pada pengaturan Anda, skrip ini dapat ditemukan di node di lokasi yang mirip dengan: `C:\Program Files\Amazon\EKS\Start-EKSBootstrap.ps1`. Anda dapat menentukan nilai parameter kustom dengan menentukannya sebagai argumen untuk skrip bootstrap. Misalnya, Anda dapat memperbarui data pengguna di template peluncuran. Untuk informasi selengkapnya, lihat [the section called “Data EC2 pengguna Amazon”](#).

Skrip ini mencakup parameter baris perintah berikut:

- `-EKSClusterName`— Menentukan nama cluster Amazon EKS untuk node pekerja ini untuk bergabung.
- `-KubeletExtraArgs`- Menentukan argumen tambahan untuk kubelet (opsional).
- `-KubeProxyExtraArgs`- Menentukan argumen tambahan untuk kube-proxy (opsional).
- `-APIServerEndpoint`— Menentukan titik akhir server API cluster Amazon EKS (opsional). Hanya berlaku bila digunakan dengan `-Base64ClusterCA`. Melewati panggilan `Get-EKSCluster`.
- `-Base64ClusterCA`- Menentukan base64 dikodekan konten CA cluster (opsional). Hanya berlaku bila digunakan dengan `-APIServerEndpoint`. Melewati panggilan `Get-EKSCluster`.
- `-DNSClusterIP`— Mengganti alamat IP yang akan digunakan untuk kueri DNS dalam cluster (opsional). Default ke `10.100.0.10` atau `172.20.0.10` berdasarkan alamat IP antarmuka utama.
- `-ServiceCIDR`— Mengganti rentang alamat IP layanan Kubernetes dari mana layanan kluster ditangani. Default ke `172.20.0.0/16` atau `10.100.0.0/16` berdasarkan alamat IP antarmuka utama.
- `-ExcludedSnatCIDRs`— Daftar IPv4 CIDRs untuk dikecualikan dari Source Network Address Translation (SNAT). Ini berarti bahwa IP pribadi pod yang dapat dialamatkan VPC tidak akan diterjemahkan ke alamat IP dari IPv4 alamat utama ENI instance untuk lalu lintas keluar. Secara default, IPv4 CIDR dari VPC untuk node Amazon EKS Windows ditambahkan. Menentukan CIDRs parameter ini juga tidak termasuk yang ditentukan. CIDRs Untuk informasi selengkapnya, lihat [the section called “Lalu lintas keluar”](#).

Selain parameter baris perintah, Anda juga dapat menentukan beberapa parameter variabel lingkungan. Saat menentukan parameter baris perintah, itu diutamakan daripada variabel lingkungan masing-masing. Variabel lingkungan harus didefinisikan sebagai cakupan mesin (atau sistem) karena skrip bootstrap hanya akan membaca variabel cakupan mesin.

Skrip memperhitungkan variabel lingkungan berikut:

- `SERVICE_IPV4_CIDR`— Lihat parameter baris `ServiceCIDR` perintah untuk definisi.
- `EXCLUDED_SNAT_CIDRS`— Harus berupa string yang dipisahkan koma. Lihat parameter baris `ExcludedSnatCIDRs` perintah untuk definisi.

Dukungan otentikasi GMSA

Amazon EKS Windows Pods memungkinkan berbagai jenis otentikasi Akun Layanan Terkelola (GMSA) grup.

- Amazon EKS mendukung identitas domain Active Directory untuk otentikasi. Untuk informasi selengkapnya tentang GMSA yang bergabung dengan domain, lihat [Otentikasi Windows di Amazon EKS Windowspods di blog](#). AWS
- Amazon EKS menawarkan plugin yang memungkinkan node non-domain-joined Windows untuk mengambil kredensial GMSA dengan identitas pengguna portabel. Untuk informasi selengkapnya tentang GMSA tanpa domain, lihat [Otentikasi Windows Tanpa Domain untuk Amazon EKS Windowspods](#) di blog. AWS

Gambar kontainer yang di-cache

Amazon EKS Windows yang dioptimalkan AMIs memiliki gambar kontainer tertentu yang di-cache untuk containerd runtime. Gambar kontainer di-cache saat membuat kustom AMIs menggunakan komponen build yang dikelola Amazon. Untuk informasi selengkapnya, lihat [the section called “Menggunakan komponen build yang dikelola Amazon”](#).

Gambar kontainer cache berikut adalah untuk containerd runtime:

- `amazonaws.com/eks/pause-windows`
- `mcr.microsoft.com/windows/nanoserver`
- `mcr.microsoft.com/windows/servercore`

Informasi lain

Untuk informasi selengkapnya tentang menggunakan Windows yang dioptimalkan Amazon EKS AMIs, lihat bagian berikut:

- Untuk detail tentang menjalankan beban kerja di Amazon EKS yang dioptimalkan untuk Windows akselerasi AMIs, lihat [the section called “Jalankan Windows GPU AMIs”](#).
- Untuk menggunakan Windows dengan grup node terkelola, lihat [the section called “Grup simpul terkelola”](#).
- Untuk meluncurkan node Windows yang dikelola sendiri, lihat [the section called “Windows”](#).
- Untuk informasi versi, lihat [the section called “Dapatkan informasi versi”](#).

- Untuk mengambil Windows Amazon EKS terbaru IDs yang dioptimalkan AMIs, lihat [the section called “Dapatkan yang terbaru IDs”](#).
- Untuk menggunakan Amazon EC2 Image Builder untuk membuat Windows yang dioptimalkan Amazon EKS kustom AMIs, lihat [the section called “Build kustom”](#).
- Untuk praktik terbaik, lihat [Manajemen AMI Windows Amazon EKS yang dioptimalkan](#) di Panduan Praktik Terbaik EKS.

Buat node Windows Server 2022 yang dikelola sendiri dengan **eksctl**

Anda dapat menggunakan yang berikut ini `test-windows-2022.yaml` sebagai referensi untuk membuat node Windows Server 2022 yang dikelola sendiri. Ganti setiap *example value* dengan nilai-nilai Anda sendiri.

Note

Anda harus menggunakan `eksctl` versi [0.116.0](#) atau yang lebih baru untuk menjalankan node Windows Server 2022 yang dikelola sendiri.

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: windows-2022-cluster
  region: region-code
  version: '1.33'

nodeGroups:
  - name: windows-ng
    instanceType: m5.2xlarge
    amiFamily: WindowsServer2022FullContainer
    volumeSize: 100
    minSize: 2
    maxSize: 3
  - name: linux-ng
    amiFamily: AmazonLinux2
    minSize: 2
    maxSize: 3
```


Grup node kemudian dapat dibuat menggunakan perintah berikut.

```
eksctl create cluster -f test-windows-2022.yaml
```

Ambil informasi versi Windows AMI

[Topik ini mencantumkan versi Windows Amazon EKS yang dioptimalkan AMIs dan versi kubelet, containerd, dan csi-proxy yang sesuai.](#)

Metadata AMI yang dioptimalkan oleh Amazon EKS, mencakup ID AMI, untuk setiap varian bisa didapatkan secara terprogram. Untuk informasi selengkapnya, lihat [the section called “Dapatkan yang terbaru IDs”](#).

AMIs diversi oleh versi Kubernetes dan tanggal rilis AMI dalam format berikut:

```
k8s_major_version.k8s_minor_version-release_date
```

Note

Grup simpul terkelola Amazon EKS mendukung rilis Windows November 2022 dan yang lebih baru AMIs.

Untuk menerima pemberitahuan semua perubahan file sumber ke halaman dokumentasi khusus ini, Anda dapat berlangganan URL berikut dengan pembaca RSS:

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/nodes/eks-ami-versions-windows.adoc.atom
```

Amazon EKS mengoptimalkan Windows Server 2022 Core AMI

Tabel berikut mencantumkan versi saat ini dan sebelumnya dari Amazon EKS yang dioptimalkan Windows Server 2022 Core AMI.

Example

Kubernetes versi 1.33

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.33-2025.06.13	1.33.1	1.7.27	1.2.1	
1.33-2025.05.17	1.33.1	1.7.27	1.2.1	

Kubernetes versi 1.32

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.32-2025.06.13	1.32.5	1.7.27	1.2.1	Ditingkatkan containerd ke 1.7.27.
1.32-2025.05.17	1.32.5	1.7.20	1.2.1	
1.32-2025.04.14	1.32.1	1.7.20	1.1.3	
1.32-2025.03.14	1.32.1	1.7.20	1.1.3	
1.32-2025.02.18	1.32.1	1.7.20	1.1.3	
1.32-2025.01.15	1.32.0	1.7.20	1.1.3	

Kubernetes versi 1.31

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.31-2025 .06.13	1.31.9	1.7.27	1.2.1	Ditingkatkan containerd ke1.7.27.
1.31-2025 .05.17	1.31.9	1.7.20	1.2.1	
1.31-2025 .04.14	1.31.5	1.7.20	1.1.3	
1.31-2025 .03.14	1.31.5	1.7.20	1.1.3	
1.31-2025 .02.15	1.31.5	1.7.20	1.1.3	
1.31-2025 .01.15	1.31.4	1.7.20	1.1.3	
1.31-2025 .01.01	1.31.4	1.7.20	1.1.3	Termasuk tambalan untukCVE-2024- 9042 .
1.31-2024 .12.13	1.31.3	1.7.20	1.1.3	
1.31-2024 .11.12	1.31.1	1.7.20	1.1.3	
1.31-2024 .10.08	1.31.1	1.7.20	1.1.3	
1.31-2024 .10.01	1.31.1	1.7.20	1.1.3	

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.31-2024 .09.10	1.31.0	1.7.20	1.1.3	

Kubernetes versi 1.30

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.30-2025 .06.13	1.30.13	1.7.27	1.2.1	Ditingkatkan containerd ke1.7.27.
1.30-2025 .05.17	1.30.13	1.7.20	1.2.1	
1.30-2025 .04.14	1.30.9	1.7.20	1.1.3	
1.30-2025 .03.14	1.30.9	1.7.20	1.1.3	Ditingkatkan containerd ke1.7.20.
1.30-2025 .02.15	1.30.9	1.7.14	1.1.3	
1.30-2025 .01.15	1.30.8	1.7.14	1.1.3	
1.30-2025 .01.01	1.30.8	1.7.14	1.1.3	Termasuk tambalan untukCVE-2024- 9042 .
1.30-2024 .12.11	1.30.7	1.7.14	1.1.3	

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.30-2024.11.12	1.30.4	1.7.14	1.1.3	
1.30-2024.10.08	1.30.4	1.7.14	1.1.3	
1.30-2024.09.10	1.30.2	1.7.14	1.1.3	
1.30-2024.08.13	1.30.2	1.7.14	1.1.3	
1.30-2024.07.10	1.30.2	1.7.14	1.1.2	Termasuk tambalan untuk CVE-2024-5321 .
1.30-2024.06.17	1.30.0	1.7.14	1.1.2	Ditingkatkan containerd ke 1.7.14.
1.30-2024.05.15	1.30.0	1.6.28	1.1.2	

Kubernetes versi 1.29

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.29-2025.06.13	1.29.15	1.7.27	1.2.1	Ditingkatkan containerd ke 1.7.27.
1.29-2025.05.17	1.29.15	1.7.20	1.2.1	

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.29-2025 .04.14	1.29.13	1.7.20	1.1.3	
1.29-2025 .03.14	1.29.13	1.7.20	1.1.3	Upgrade containerd ke1.7.20.
1.29-2025 .02.15	1.29.13	1.7.14	1.1.3	
1.29-2025 .01.15	1.29.12	1.7.14	1.1.3	
1.29-2025 .01.01	1.29.12	1.7.14	1.1.3	Termasuk tambalan untukCVE-2024- 9042 .
1.29-2024 .12.11	1.29.10	1.7.14	1.1.3	
1.29-2024 .11.12	1.29.8	1.7.14	1.1.3	
1.29-2024 .10.08	1.29.8	1.7.14	1.1.3	
1.29-2024 .09.10	1.29.6	1.7.14	1.1.3	
1.29-2024 .08.13	1.29.6	1.7.14	1.1.3	
1.29-2024 .07.10	1.29.6	1.7.11	1.1.2	Termasuk tambalan untukCVE-2024- 5321 .

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.29-2024.06.17	1.29.3	1.7.11	1.1.2	
1.29-2024.05.15	1.29.3	1.7.11	1.1.2	Upgrade containerd ke1.7.11. Upgrade kubelet ke1.29.3.
1.29-2024.04.09	1.29.0	1.6.28	1.1.2	Upgrade containerd ke1.6.28. Membangun kembali CNI dan csi-proxy menggunakan golang 1.22.1
1.29-2024.03.12	1.29.0	1.6.25	1.1.2	
1.29-2024.02.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.06	1.29.0	1.6.25	1.1.2	Memperbaiki bug di mana gambar jeda salah dihapus oleh proses pengumpulan kubelet sampah.

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.29-2024 .01.11	1.29.0	1.6.18	1.1.2	Dikecualikan Pembaruan Windows Mandiri KB5034439 di Windows Server 2022 Core. AMIs KB hanya berlaku untuk instalasi Windows dengan partisi WinRE terpisah, yang tidak disertakan dengan Windows Amazon EKS Optimized kami. AMIs

Kubernetes versi 1.28

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.28-2025 .06.13	1.28.15	1.7.27	1.2.1	Upgrade containerd ke 1.7.27.
1.28-2025 .05.17	1.28.15	1.7.20	1.2.1	
1.28-2025 .04.14	1.28.15	1.7.20	1.1.3	

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.28-2025 .03.14	1.28.15	1.7.20	1.1.3	Upgrade containerd ke1.7.20.
1.28-2025 .02.15	1.28.15	1.7.14	1.1.3	
1.28-2025 .01.15	1.28.15	1.7.14	1.1.3	
1.28-2025 -01-01	1.28.15	1.7.14	1.1.3	Termasuk tambalan untukCVE-2024- 9042 .
1.28-2024 .12.11	1.28.15	1.7.14	1.1.3	
1.28-2024 .11.12	1.28.13	1.7.14	1.1.3	
1.28-2024 .10.08	1.28.13	1.7.14	1.1.3	
1.28-2024 .09.10	1.28.11	1.7.14	1.1.3	
1.28-2024 .08.13	1.28.11	1.7.14	1.1.3	
1.28-2024 .07.10	1.28.11	1.7.11	1.1.2	Termasuk tambalan untukCVE-2024- 5321 .

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.28-2024.06.17	1.28.8	1.7.11	1.1.2	Upgrade containerd ke 1.7.11.
1.28-2024.05.14	1.28.8	1.6.28	1.1.2	Upgrade containerd ke 1.6.28. Upgrade kubelet ke 1.28.8.
1.28-2024.04.09	1.28.5	1.6.25	1.1.2	Upgrade containerd ke 1.6.25. Membangun kembali CNI dan csi-proxy menggunakan golang 1.22.1
1.28-2024.03.12	1.28.5	1.6.18	1.1.2	
1.28-2024.02.13	1.28.5	1.6.18	1.1.2	

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.28-2024 .01.11	1.28.5	1.6.18	1.1.2	Dikecualikan Pembaruan Windows Mandiri KB5034439 di Windows Server 2022 Core. AMIs KB hanya berlaku untuk instalasi Windows dengan partisi WinRE terpisah, yang tidak disertaka n dengan Windows Amazon EKS Optimized kami. AMIs
1.28-2023 .12.12	1.28.3	1.6.18	1.1.2	
1.28-2023 .11.14	1.28.3	1.6.18	1.1.2	Termasuk tambalan untuk CVE-2023- 5528 .

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.28-2023 .10.19	1.28.2	1.6.18	1.1.2	Upgrade containerd ke 1.6.18. Ditambahkan variabel lingkungan skrip bootstrap baru (SERVICE_IPV4_CIDR dan EXCLUDE_SNAT_CIDR S).
1.28-2023 -09.27	1.28.2	1.6.6	1.1.2	Memperbaiki penasihat keamanan di kubelet.
1.28-2023 .09.12	1.28.1	1.6.6	1.1.2	

Kubernetes versi 1.27

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.27-2025 .06.13	1.27.16	1.7.27	1.2.1	Upgrade containerd ke 1.7.27.
1.27-2025 .05.17	1.27.16	1.7.20	1.2.1	
1.27-2025 .04.14	1.27.16	1.7.20	1.1.3	

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.27-2025 .03.14	1.27.16	1.7.20	1.1.3	Upgrade containerd ke1.7.20.
1.27-2025 .02.15	1.27.16	1.7.14	1.1.3	
1.27-2025 .01.15	1.27.16	1.7.14	1.1.3	
1.27-2025 .01.01	1.27.16	1.7.14	1.1.3	Termasuk tambalan untukCVE-2024- 9042 .
1.27-2024 .12.11	1.27.16	1.7.14	1.1.3	
1.27-2024 .11.12	1.27.16	1.7.14	1.1.3	
1.27-2024 .10.08	1.27.16	1.7.14	1.1.3	
1.27-2024 .09.10	1.27.15	1.7.14	1.1.3	
1.27-2024 .08.13	1.27.15	1.7.14	1.1.3	
1.27-2024 .07.10	1.27.15	1.7.11	1.1.2	Termasuk tambalan untukCVE-2024- 5321 .

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.27-2024.06.17	1.27.12	1.7.11	1.1.2	Upgrade containerd ke 1.7.11.
1.27-2024.05.14	1.27.12	1.6.28	1.1.2	Upgrade containerd ke 1.6.28. Upgrade kubelet ke 1.27.12.
1.27-2024.04.09	1.27.9	1.6.25	1.1.2	Upgrade containerd ke 1.6.25. Membangun kembali CNI dan csi-proxy menggunakan golang 1.22.1
1.27-2024.03.12	1.27.9	1.6.18	1.1.2	
1.27-2024.02.13	1.27.9	1.6.18	1.1.2	

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.27-2024 .01.11	1.27.9	1.6.18	1.1.2	Dikecualikan Pembaruan Windows Mandiri KB5034439 di Windows Server 2022 Core. AMIs KB hanya berlaku untuk instalasi Windows dengan partisi WinRE terpisah, yang tidak disertaka n dengan Windows Amazon EKS Optimized kami. AMIs
1.27-2023 .12.12	1.27.7	1.6.18	1.1.2	
1.27-2023 .11.14	1.27.7	1.6.18	1.1.2	Termasuk tambalan untuk CVE-2023- 5528 .

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.27-2023.10.19	1.27.6	1.6.18	1.1.2	Upgrade containerd ke 1.6.18. Ditambahkan variabel lingkungan skrip bootstrap baru (SERVICE_IPV4_CIDR dan EXCLUDE_SNAT_CIDR S).
1.27-2023-09.27	1.27.6	1.6.6	1.1.2	Memperbaiki penasihat keamanan di kubelet.
1.27-2023.09.12	1.27.4	1.6.6	1.1.2	Upgrade plugin Amazon VPC CNI untuk menggunakan biner konektor Kubernetes, yang mendapatkan alamat IP Pod dari server API Kubernetes. Permintaan tarik gabungan #100 .

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.27-2023.08.17	1.27.4	1.6.6	1.1.2	Termasuk tambalan untuk CVE-2023-3676 , CVE-2023-3893 , dan CVE-2023-3955 .
1.27-2023.08.08	1.27.3	1.6.6	1.1.1	
1.27-2023.07.11	1.27.3	1.6.6	1.1.1	
1.27-2023.06.20	1.27.1	1.6.6	1.1.1	Masalah teratasi yang menyebabkan daftar pencarian akhiran DNS salah diisi.
1.27-2023.06.14	1.27.1	1.6.6	1.1.1	Menambahk an dukungan untuk pemetaan port host di CNI. Permintaan tarik gabungan #93 .

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.27-2023 .06.06	1.27.1	1.6.6	1.1.1	Memperbaiki container s-roadmap masalah #2042 , yang menyebabkan node gagal menarik gambar ECR Amazon pribadi.
1.27-2023 .05.17	1.27.1	1.6.6	1.1.1	

Kubernetes versi 1.26

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.26-2025 .05.17	1.26.15	1.7.20	1.2.1	
1.26-2025 .04.14	1.26.15	1.7.20	1.1.3	
1.26-2025 .03.14	1.26.15	1.7.20	1.1.3	Upgrade containerd ke1.7.20.
1.26-2025 .02.15	1.26.15	1.7.14	1.1.3	
1.26-2025 .01.15	1.26.15	1.7.14	1.1.3	

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.26-2024 .12.11	1.26.15	1.7.14	1.1.3	
1.26-2024 .11.12	1.26.15	1.7.14	1.1.3	
1.26-2024 .10.08	1.26.15	1.7.14	1.1.3	
1.26-2024 .09.10	1.26.15	1.7.14	1.1.3	
1.26-2024 .08.13	1.26.15	1.7.14	1.1.3	
1.26-2024 .07.10	1.26.15	1.7.11	1.1.2	Termasuk tambalan untuk CVE-2024- 5321 .
1.26-2024 .06.17	1.26.15	1.7.11	1.1.2	Upgrade containerd ke 1.7.11.
1.26-2024 .05.14	1.26.15	1.6.28	1.1.2	Upgrade container d ke 1.6.28. Upgrade kubelet ke 1.26.15.

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.26-2024 .04.09	1.26.12	1.6.25	1.1.2	Upgrade container d ke1.6.25. Membangun kembali CNI dan csi-proxy menggunakan golang 1.22.1
1.26-2024 .03.12	1.26.12	1.6.18	1.1.2	
1.26-2024 .02.13	1.26.12	1.6.18	1.1.2	

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.26-2024 .01.11	1.26.12	1.6.18	1.1.2	Dikecualikan Pembaruan Windows Mandiri KB5034439 di Windows Server 2022 Core. AMIs KB hanya berlaku untuk instalasi Windows dengan partisi WinRE terpisah, yang tidak disertaka n dengan Windows Amazon EKS Optimized kami. AMIs
1.26-2023 .12.12	1.26.10	1.6.18	1.1.2	
1.26-2023 .11.14	1.26.10	1.6.18	1.1.2	Termasuk tambalan untuk CVE-2023- 5528 .

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.26-2023.10.19	1.26.9	1.6.18	1.1.2	Upgrade containerd ke 1.6.18. Upgrade kubelet ke 1.26.9. Ditambahkan variabel lingkungan skrip bootstrap baru (SERVICE_IP, PV4_CIDR, dan EXCLUDE_SNAT_CIDR S).
1.26-2023.09.12	1.26.7	1.6.6	1.1.2	Upgrade plugin Amazon VPC CNI untuk menggunakan biner konektor Kubernetes, yang mendapatkan alamat IP Pod dari server API Kubernetes. Permintaan tarik gabungan #100 .
1.26-2023.08.17	1.26.7	1.6.6	1.1.2	Termasuk tambalan untuk CVE-2023-3676, CVE-2023-3893, dan CVE-2023-3955.

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.26-2023 .08.08	1.26.6	1.6.6	1.1.1	
1.26-2023 .07.11	1.26.6	1.6.6	1.1.1	
1.26-2023 .06.20	1.26.4	1.6.6	1.1.1	Masalah teratasi yang menyebabkan daftar pencarian akhiran DNS salah diisi.
1.26-2023 .06.14	1.26.4	1.6.6	1.1.1	Kubernetes yang ditingkatkan menjadi. 1.26.4 Menambahkan dukungan untuk pemetaan port host di CNI. Permintaan tarik gabungan #93 .

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.26-2023 .05.09	1.26.2	1.6.6	1.1.1	Memperbaiki bug yang menyebabkan masalah konektivitas jaringan #1126 pada pod setelah node restart. Memperkenalkan parameter konfigurasi skrip bootstrap baru (ExcludedS natCIDRs).
1.26-2023 .04.26	1.26.2	1.6.6	1.1.1	
1.26-2023 .04.11	1.26.2	1.6.6	1.1.1	Menambahkan mekanisme pemulihan untuk kubelet dan kube-proxy pada kerusakan layanan.
1.26-2023 .03.24	1.26.2	1.6.6	1.1.1	

Amazon EKS mengoptimalkan Windows Server 2022 AMI Penuh

Tabel berikut mencantumkan versi saat ini dan sebelumnya dari Amazon EKS yang dioptimalkan Windows Server 2022 Full AMI.

Example

Kubernetes versi 1.33

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.33-2025.06.13	1.33.1	1.7.27	1.2.1	
1.33-2025.05.17	1.33.1	1.7.27	1.2.1	

Kubernetes versi 1.32

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.32-2025.06.13	1.32.5	1.7.27	1.2.1	Upgrade ke containerd 1.7.27
1.32-2025.05.17	1.32.5	1.7.20	1.2.1	
1.32-2025.04.14	1.32.1	1.7.20	1.1.3	
1.32-2025.03.14	1.32.1	1.7.20	1.1.3	
1.32-2025.02.18	1.32.1	1.7.20	1.1.3	
1.32-2025.01.01	1.32.0	1.7.20	1.1.3	

Kubernetes versi 1.31

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.31-2025 .06.13	1.31.9	1.7.27	1.2.1	Upgrade containerd ke1.7.27.
1.31-2025 .05.17	1.31.9	1.7.20	1.2.1	
1.31-2025 .04.14	1.31.5	1.7.20	1.1.3	
1.31-2025 .03.14	1.31.5	1.7.20	1.1.3	
1.31-2025 .02.15	1.31.5	1.7.20	1.1.3	
1.31-2025 .01.15	1.31.4	1.7.20	1.1.3	
1.31-2025 .01.01	1.31.4	1.7.20	1.1.3	Termasuk tambalan untukCVE-2024- 9042 .
1.31-2024 .12.13	1.31.3	1.7.20	1.1.3	
1.31-2024 .11.12	1.31.1	1.7.20	1.1.3	
1.31-2024 .10.08	1.31.1	1.7.20	1.1.3	
1.31-2024 .10.01	1.31.1	1.7.20	1.1.3	

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.31-2024 .09.10	1.31.0	1.7.20	1.1.3	

Kubernetes versi 1.30

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.30-2025 .06.13	1.30.13	1.7.27	1.2.1	Upgrade containerd ke1.7.27.
1.30-2025 .05.17	1.30.13	1.7.20	1.2.1	
1.30-2025 .04.14	1.30.9	1.7.20	1.1.3	
1.30-2025 .03.14	1.30.9	1.7.20	1.1.3	Upgrade containerd ke1.7.20.
1.30-2025 .02.15	1.30.9	1.7.14	1.1.3	
1.30-2025 .01.15	1.30.8	1.7.14	1.1.3	
1.30-2025 .01.01	1.30.8	1.7.14	1.1.3	Termasuk tambalan untukCVE-2024- 9042 .
1.30-2024 .12.11	1.30.7	1.7.14	1.1.3	

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.30-2024 .11.12	1.30.4	1.7.14	1.1.3	
1.30-2024 .10.08	1.30.4	1.7.14	1.1.3	
1.30-2024 .09.10	1.30.2	1.7.14	1.1.3	
1.30-2024 .08.13	1.30.2	1.7.14	1.1.3	
1.30-2024 .07.10	1.30.2	1.7.14	1.1.2	Termasuk tambalan untuk CVE-2024- 5321 .
1.30-2024 .06.17	1.30.0	1.7.14	1.1.2	Upgrade containerd ke 1.7.14.
1.30-2024 .05.15	1.30.0	1.6.28	1.1.2	

Kubernetes versi 1.29

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.29-2025 .06.13	1.29.15	1.7.27	1.2.1	Upgrade containerd ke 1.7.27.
1.29-2025 .05.17	1.29.15	1.7.20	1.2.1	

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.29-2025 .04.14	1.29.13	1.7.20	1.1.3	
1.29-2025 .03.14	1.29.13	1.7.20	1.1.3	Upgrade containerd ke1.7.20.
1.29-2025 .02.15	1.29.13	1.7.14	1.1.3	
1.29-2025 .01.15	1.29.12	1.7.14	1.1.3	
1.29-2025 .01.01	1.29.12	1.7.14	1.1.3	Termasuk tambalan untukCVE-2024- 9042 .
1.29-2024 .12.11	1.29.10	1.7.14	1.1.3	
1.29-2024 .11.12	1.29.8	1.7.14	1.1.3	
1.29-2024 .10.08	1.29.8	1.7.14	1.1.3	
1.29-2024 .09.10	1.29.6	1.7.14	1.1.3	
1.29-2024 .08.13	1.29.6	1.7.14	1.1.3	
1.29-2024 .07.10	1.29.6	1.7.11	1.1.2	Termasuk tambalan untukCVE-2024- 5321 .

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.29-2024.06.17	1.29.3	1.7.11	1.1.2	
1.29-2024.05.15	1.29.3	1.7.11	1.1.2	Upgrade containerd ke 1.7.11. Upgrade kubelet ke 1.29.3.
1.29-2024.04.09	1.29.0	1.6.28	1.1.2	Upgrade containerd ke 1.6.28. Membangun kembali CNI dan csi-proxy menggunakan golang 1.22.1
1.29-2024.03.12	1.29.0	1.6.25	1.1.2	
1.29-2024.02.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.06	1.29.0	1.6.25	1.1.2	Memperbaiki bug di mana gambar jeda salah dihapus oleh proses pengumpulan kubelet sampah.

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.29-2024 .01.09	1.29.0	1.6.18	1.1.2	

Kubernetes versi 1.28

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.28-2025 .06.13	1.28.15	1.7.27	1.2.1	Upgrade containerd ke1.7.27.
1.28-2025 .05.17	1.28.15	1.7.20	1.2.1	
1.28-2025 .04.14	1.28.15	1.7.20	1.1.3	
1.28-2025 .03.14	1.28.15	1.7.20	1.1.3	Upgrade containerd ke1.7.20.
1.28-2025 .02.15	1.28.15	1.7.14	1.1.3	
1.28-2025 .01.15	1.28.15	1.7.14	1.1.3	
1.28-2025 .01.01	1.28.15	1.7.14	1.1.3	Termasuk tambalan untukCVE-2024- 9042 .
1.28-2024 .12.11	1.28.15	1.7.14	1.1.3	

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.28-2024 .11.12	1.28.13	1.7.14	1.1.3	
1.28-2024 .10.08	1.28.13	1.7.14	1.1.3	
1.28-2024 .09.10	1.28.11	1.7.14	1.1.3	
1.28-2024 .08.13	1.28.11	1.7.14	1.1.3	
1.28-2024 .07.10	1.28.11	1.7.11	1.1.2	Termasuk tambalan untuk CVE-2024- 5321 .
1.28-2024 .06.17	1.28.8	1.7.11	1.1.2	Upgrade containerd ke 1.7.11.
1.28-2024 .05.14	1.28.8	1.6.28	1.1.2	Upgrade container d ke 1.6.28. Upgrade kubelet ke 1.28.8.

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.28-2024 .04.09	1.28.5	1.6.25	1.1.2	Upgrade container d ke1.6.25. Membangun kembali CNI dan csi-proxy menggunak an. golang 1.22.1
1.28-2024 .03.12	1.28.5	1.6.18	1.1.2	
1.28-2024 .02.13	1.28.5	1.6.18	1.1.2	
1.28-2024 .01.09	1.28.5	1.6.18	1.1.2	
1.28-2023 .12.12	1.28.3	1.6.18	1.1.2	
1.28-2023 .11.14	1.28.3	1.6.18	1.1.2	Termasuk tambalan untukCVE-2023- 5528 .

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.28-2023 .10.19	1.28.2	1.6.18	1.1.2	Upgrade containerd ke 1.6.18. Ditambahkan variabel lingkungan skrip bootstrap baru (SERVICE_IPV4_CIDR dan EXCLUDE_SNAT_CIDR S).
1.28-2023 -09.27	1.28.2	1.6.6	1.1.2	Memperbaiki penasihat keamanan di kubelet.
1.28-2023 .09.12	1.28.1	1.6.6	1.1.2	

Kubernetes versi 1.27

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.27-2025 .06.13	1.27.16	1.7.27	1.2.1	Upgrade containerd ke 1.7.27.
1.27-2025 .05.17	1.27.16	1.7.20	1.2.1	
1.27-2025 .04.14	1.27.16	1.7.20	1.1.3	

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.27-2025 .03.14	1.27.16	1.7.20	1.1.3	Upgrade containerd ke1.7.20.
1.27-2025 .02.15	1.27.16	1.7.14	1.1.3	
1.27-2025 .01.15	1.27.16	1.7.14	1.1.3	
1.27-2025 .01.01	1.27.16	1.7.14	1.1.3	Termasuk tambalan untukCVE-2024- 9042 .
1.27-2024 .12.11	1.27.16	1.7.14	1.1.3	
1.27-2024 .11.12	1.27.16	1.7.14	1.1.3	
1.27-2024 .10.08	1.27.16	1.7.14	1.1.3	
1.27-2024 .09.10	1.27.15	1.7.14	1.1.3	
1.27-2024 .08.13	1.27.15	1.7.14	1.1.3	
1.27-2024 .07.10	1.27.15	1.7.11	1.1.2	Termasuk tambalan untukCVE-2024- 5321 .

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.27-2024 .06.17	1.27.12	1.7.11	1.1.2	Upgrade containerd ke1.7.11.
1.27-2024 .05.14	1.27.12	1.6.28	1.1.2	Upgrade container d ke1.6.28. Upgrade kubelet ke1.27.12.
1.27-2024 .04.09	1.27.9	1.6.25	1.1.2	Upgrade container d ke1.6.25. Membangun kembali CNI dan csi-proxy menggunakan golang 1.22.1
1.27-2024 .03.12	1.27.9	1.6.18	1.1.2	
1.27-2024 .02.13	1.27.9	1.6.18	1.1.2	
1.27-2024 .01.09	1.27.9	1.6.18	1.1.2	
1.27-2023 .12.12	1.27.7	1.6.18	1.1.2	

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.27-2023 .11.14	1.27.7	1.6.18	1.1.2	Termasuk tambalan untuk CVE-2023- 5528 .
1.27-2023 .10.19	1.27.6	1.6.18	1.1.2	Upgrade container d ke 1.6.18. Ditambahk an variabel lingkungan skrip bootstrap baru (SERVICE_I PV4_CIDR dan EXCLUDE SNAT_CIDR S).
1.27-2023 -09.27	1.27.6	1.6.6	1.1.2	Memperbai ki penasihat keamanan di kubelet.
1.27-2023 .09.12	1.27.4	1.6.6	1.1.2	Upgrade plugin Amazon VPC CNI untuk menggunakan biner konektor Kubernetes, yang mendapatk an alamat IP Pod dari server API Kubernetes. Permintaan tarik gabungan #100 .

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.27-2023 .08.17	1.27.4	1.6.6	1.1.2	Termasuk tambalan untuk CVE-2023- 3676 , CVE-2023- 3893 , dan CVE-2023- 3955 .
1.27-2023 .08.08	1.27.3	1.6.6	1.1.1	
1.27-2023 .07.11	1.27.3	1.6.6	1.1.1	
1.27-2023 .06.20	1.27.1	1.6.6	1.1.1	Masalah teratasi yang menyebabkan daftar pencarian akhiran DNS salah diisi.
1.27-2023 .06.14	1.27.1	1.6.6	1.1.1	Menambahk an dukungan untuk pemetaan port host di CNI. Permintaan tarik gabungan #93.

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.27-2023 .06.06	1.27.1	1.6.6	1.1.1	Memperbaiki container s-roadmap masalah #2042 , yang menyebabkan node gagal menarik gambar ECR Amazon pribadi.
1.27-2023 .05.18	1.27.1	1.6.6	1.1.1	

Kubernetes versi 1.26

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.26-2025 .05.17	1.26.15	1.7.20	1.2.1	
1.26-2025 .04.14	1.26.15	1.7.20	1.1.3	
1.26-2025 .03.14	1.26.15	1.7.20	1.1.3	Upgrade containerd ke1.7.20.
1.26-2025 .02.15	1.26.15	1.7.14	1.1.3	
1.26-2025 .01.15	1.26.15	1.7.14	1.1.3	

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.26-2024 .12.11	1.26.15	1.7.14	1.1.3	
1.26-2024 .11.12	1.26.15	1.7.14	1.1.3	
1.26-2024 .10.08	1.26.15	1.7.14	1.1.3	
1.26-2024 .09.10	1.26.15	1.7.14	1.1.3	
1.26-2024 .08.13	1.26.15	1.7.14	1.1.3	
1.26-2024 .07.10	1.26.15	1.7.11	1.1.2	Termasuk tambalan untuk CVE-2024- 5321 .
1.26-2024 .06.17	1.26.15	1.7.11	1.1.2	Upgrade containerd ke 1.7.11.
1.26-2024 .05.14	1.26.15	1.6.28	1.1.2	Upgrade container d ke 1.6.28. Upgrade kubelet ke 1.26.15.

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.26-2024 .04.09	1.26.12	1.6.25	1.1.2	Upgrade container d ke1.6.25. Membangun kembali CNI dan csi-proxy menggunakan. golang 1.22.1
1.26-2024 .03.12	1.26.12	1.6.18	1.1.2	
1.26-2024 .02.13	1.26.12	1.6.18	1.1.2	
1.26-2024 .01.09	1.26.12	1.6.18	1.1.2	
1.26-2023 .12.12	1.26.10	1.6.18	1.1.2	
1.26-2023 .11.14	1.26.10	1.6.18	1.1.2	Termasuk tambalan untukCVE-2023- 5528 .

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.26-2023.10.19	1.26.9	1.6.18	1.1.2	Upgrade containerd ke 1.6.18. Upgrade kubelet ke 1.26.9. Ditambahkan variabel lingkungan skrip bootstrap baru (SERVICE_IP, PV4_CIDR, dan EXCLUDE_SNAT_CIDR S).
1.26-2023.09.12	1.26.7	1.6.6	1.1.2	Upgrade plugin Amazon VPC CNI untuk menggunakan biner konektor Kubernetes, yang mendapatkan alamat IP Pod dari server API Kubernetes. Permintaan tarik gabungan #100 .
1.26-2023.08.17	1.26.7	1.6.6	1.1.2	Termasuk tambalan untuk CVE-2023-3676, CVE-2023-3893, dan CVE-2023-3955.

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.26-2023.08.08	1.26.6	1.6.6	1.1.1	
1.26-2023.07.11	1.26.6	1.6.6	1.1.1	
1.26-2023.06.20	1.26.4	1.6.6	1.1.1	Masalah teratasi yang menyebabkan daftar pencarian akhiran DNS salah diisi.
1.26-2023.06.14	1.26.4	1.6.6	1.1.1	Kubernetes yang ditingkatkan menjadi. 1.26.4 Menambahkan dukungan untuk pemetaan port host di CNI. Permintaan tarik gabungan #93 .

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.26-2023 .05.09	1.26.2	1.6.6	1.1.1	Memperbaiki bug yang menyebabkan masalah konektivitas jaringan #1126 pada pod setelah node restart. Memperkenalkan parameter konfigurasi skrip bootstrap baru (ExcludedS natCIDRs).
1.26-2023 .04.26	1.26.2	1.6.6	1.1.1	
1.26-2023 .04.11	1.26.2	1.6.6	1.1.1	Menambahkan mekanisme pemulihan untuk kubelet dan kube-proxy pada kerusakan layanan.
1.26-2023 .03.24	1.26.2	1.6.6	1.1.1	

Windows Server 2019 Core AMI yang dioptimalkan oleh Amazon EKS

Tabel berikut mencantumkan versi saat ini dan sebelumnya dari Amazon EKS yang dioptimalkan Windows Server 2019 Core AMI.

Example

Kubernetes versi 1.33

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.33-2025.06.13	1.33.1	1.7.27	1.2.1	
1.33-2025.05.17	1.33.1	1.7.27	1.2.1	

Kubernetes versi 1.32

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.32-2025.06.13	1.32.5	1.7.27	1.2.1	Upgrade containerd ke 1.7.27.
1.32-2025.05.17	1.32.5	1.7.20	1.2.1	
1.32-2025.04.14	1.32.1	1.7.20	1.1.3	
1.32-2025.03.14	1.32.1	1.7.20	1.1.3	
1.32-2025.02.18	1.32.1	1.7.20	1.1.3	
1.32-2025.01.15	1.32.4	1.7.20	1.1.3	

Kubernetes versi 1.31

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.31-2025 .06.13	1.31.9	1.7.27	1.2.1	Upgrade containerd ke1.7.27.
1.31-2025 .05.17	1.31.9	1.7.20	1.2.1	
1.31-2025 .04.14	1.31.5	1.7.20	1.1.3	
1.31-2025 .03.14	1.31.5	1.7.20	1.1.3	
1.31-2025 .02.15	1.31.5	1.7.20	1.1.3	
1.31-2025 .01.15	1.31.4	1.7.20	1.1.3	
1.31-2025 .01.01	1.31.4	1.7.20	1.1.3	Termasuk tambalan untukCVE-2024- 9042 .
1.31-2024 .12.13	1.31.3	1.7.20	1.1.3	
1.31-2024 .11.12	1.31.1	1.7.20	1.1.3	
1.31-2024 .10.08	1.31.1	1.7.20	1.1.3	
1.31-2024 .10.01	1.31.1	1.7.20	1.1.3	

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.31-2024 .09.10	1.31.0	1.7.20	1.1.3	

Kubernetes versi 1.30

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.30-2025 .06.13	1.30.13	1.7.27	1.2.1	Upgrade containerd ke1.7.27.
1.30-2025 .05.17	1.30.13	1.7.20	1.2.1	
1.30-2025 .04.14	1.30.9	1.7.20	1.1.3	
1.30-2025 .03.14	1.30.9	1.7.20	1.1.3	Upgrade containerd ke1.7.20.
1.30-2025 -02-15	1.30.9	1.7.14	1.1.3	
1.30-2025 .01.15	1.30.8	1.7.14	1.1.3	
1.30-2025 .01.01	1.30.8	1.7.14	1.1.3	Termasuk tambalan untukCVE-2024- 9042 .
1.30-2024 .12.11	1.30.7	1.7.14	1.1.3	

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.30-2024.11.12	1.30.4	1.7.14	1.1.3	
1.30-2024.10.08	1.30.4	1.7.14	1.1.3	
1.30-2024.09.10	1.30.2	1.7.14	1.1.3	
1.30-2024.08.13	1.30.2	1.7.14	1.1.3	
1.30-2024.07.10	1.30.2	1.7.14	1.1.2	Termasuk tambalan untuk CVE-2024-5321 .
1.30-2024.06.17	1.30.0	1.7.14	1.1.2	Upgrade containerd ke 1.7.14.
1.30-2024.05.15	1.30.0	1.6.28	1.1.2	

Kubernetes versi 1.29

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.29-2025.06.13	1.29.15	1.7.27	1.2.1	Upgrade containerd ke 1.7.27.
1.29-2025.05.17	1.29.15	1.7.20	1.2.1	

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.29-2025 .04.14	1.29.13	1.7.20	1.1.3	
1.29-2025 .03.14	1.29.13	1.7.20	1.1.3	Upgrade containerd ke1.7.20.
1.29-2025 .02.15	1.29.13	1.7.14	1.1.3	
1.29-2025 .01.15	1.29.12	1.7.14	1.1.3	
1.29-2025 .01.01	1.29.12	1.7.14	1.1.3	Termasuk tambalan untukCVE-2024- 9042 .
1.29-2024 .12.11	1.29.10	1.7.14	1.1.3	
1.29-2024 .11.12	1.29.8	1.7.14	1.1.3	
1.29-2024 .10.08	1.29.8	1.7.14	1.1.3	
1.29-2024 .09.10	1.29.6	1.7.14	1.1.3	
1.29-2024 .08.13	1.29.6	1.7.14	1.1.3	
1.29-2024 .07.10	1.29.6	1.7.11	1.1.2	Termasuk tambalan untukCVE-2024- 5321 .

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.29-2024.06.17	1.29.3	1.7.11	1.1.2	
1.29-2024.05.15	1.29.3	1.7.11	1.1.2	Upgrade containerd ke 1.7.11. Upgrade kubelet ke 1.29.3.
1.29-2024.04.09	1.29.0	1.6.28	1.1.2	Ditingkatkan containerd ke 1.6.28. Membangun kembali CNI dan csi-proxy menggunakan golang 1.22.1
1.29-2024.03.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.06	1.29.0	1.6.25	1.1.2	Memperbaiki bug di mana gambar jeda salah dihapus oleh proses pengumpulan kubelet sampah.

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.29-2024 .01.09	1.29.0	1.6.18	1.1.2	

Kubernetes versi 1.28

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.28-2025 .06.13	1.28.15	1.7.20	1.2.1	Ditingkatkan containerd ke1.7.27.
1.28-2025 .05.17	1.28.15	1.7.20	1.2.1	
1.28-2025 .04.14	1.28.15	1.7.20	1.1.3	
1.28-2025 .03.14	1.28.15	1.7.20	1.1.3	Ditingkatkan containerd ke1.7.20.
1.28-2025 .02.15	1.28.15	1.7.14	1.1.3	
1.28-2025 -01-15	1.28.15	1.7.14	1.1.3	
1.28-2025 -01-01	1.28.15	1.7.14	1.1.3	Termasuk tambalan untukCVE-2024- 9042 .
1.28-2024 .12.11	1.28.15	1.7.14	1.1.3	

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.28-2024 .11.12	1.28.13	1.7.14	1.1.3	
1.28-2024 .10.08	1.28.13	1.7.14	1.1.3	
1.28-2024 .09.10	1.28.11	1.7.14	1.1.3	
1.28-2024 .08.13	1.28.11	1.7.14	1.1.3	
1.28-2024 .07.10	1.28.11	1.7.11	1.1.2	Termasuk tambalan untuk CVE-2024- 5321 .
1.28-2024 .06.17	1.28.8	1.7.11	1.1.2	Ditingkatkan containerd ke 1.7.11.
1.28-2024 .05.14	1.28.8	1.6.28	1.1.2	Ditingkatkan container d ke 1.6.28. Ditingkatkan kubelet ke 1.28.8.

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.28-2024 .04.09	1.28.5	1.6.25	1.1.2	Ditingkatkan containerd ke 1.6.25. Membangun kembali CNI dan csi-proxy menggunakan golang 1.22.1
1.28-2024 .03.13	1.28.5	1.6.18	1.1.2	
1.28-2024 .02.13	1.28.5	1.6.18	1.1.2	
1.28-2024 .01.09	1.28.5	1.6.18	1.1.2	
1.28-2023 .12.12	1.28.3	1.6.18	1.1.2	
1.28-2023 .11.14	1.28.3	1.6.18	1.1.2	Termasuk tambalan untuk CVE-2023-5528 .

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.28-2023 .10.19	1.28.2	1.6.18	1.1.2	Ditingkatkan containerd ke 1.6.18. Ditambahkan variabel lingkungan skrip bootstrap baru (SERVICE_IPV4_CIDR dan EXCLUDE_SNAT_CIDR S).
1.28-2023 -09.27	1.28.2	1.6.6	1.1.2	Memperbaiki penasihat keamanan di kubelet.
1.28-2023 .09.12	1.28.1	1.6.6	1.1.2	

Kubernetes versi 1.27

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.27-2025 .06.13	1.27.16	1.7.27	1.2.1	Ditingkatkan containerd ke 1.7.27.
1.27-2025 .05.17	1.27.16	1.7.20	1.2.1	
1.27-2025 .04.14	1.27.16	1.7.20	1.1.3	

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.27-2025 .03.14	1.27.16	1.7.20	1.1.3	Ditingkatkan containerd ke1.7.20.
1.27-2025 .02.15	1.27.16	1.7.14	1.1.3	
1.27-2025 .01.15	1.27.16	1.7.14	1.1.3	
1.27-2025 .01.01	1.27.16	1.7.14	1.1.3	Termasuk tambalan untukCVE-2024- 9042 .
1.27-2024 .12.11	1.27.16	1.7.14	1.1.3	
1.27-2024 .11.12	1.27.16	1.7.14	1.1.3	
1.27-2024 .10.08	1.27.16	1.7.14	1.1.3	
1.27-2024 .09.10	1.27.15	1.7.14	1.1.3	
1.27-2024 .08.13	1.27.15	1.7.14	1.1.3	
1.27-2024 .07.10	1.27.15	1.7.11	1.1.2	Termasuk tambalan untukCVE-2024- 5321 .

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.27-2024 .06.17	1.27.12	1.7.11	1.1.2	Ditingkatkan containerd ke1.7.11.
1.27-2024 .05.14	1.27.12	1.6.28	1.1.2	Ditingkatkan container d ke1.6.28. Ditingkatkan kubelet ke1.27.12.
1.27-2024 .04.09	1.27.9	1.6.25	1.1.2	Ditingkatkan container d ke1.6.25. Membangun kembali CNI dan csi-proxy menggunakan golang 1.22.1
1.27-2024 .03.13	1.27.9	1.6.18	1.1.2	
1.27-2024 .02.13	1.27.9	1.6.18	1.1.2	
1.27-2024 .01.09	1.27.9	1.6.18	1.1.2	
1.27-2023 .12.12	1.27.7	1.6.18	1.1.2	

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.27-2023 .11.14	1.27.7	1.6.18	1.1.2	Termasuk tambalan untuk CVE-2023-5528 .
1.27-2023 .10.19	1.27.6	1.6.18	1.1.2	Ditingkatkan containerd ke 1.6.18. Ditambahkan variabel lingkungan skrip bootstrap baru (SERVICE_IPV4_CIDR dan EXCLUDE_SNAT_CIDR S).
1.27-2023 -09.27	1.27.6	1.6.6	1.1.2	Memperbaiki penasihat keamanan di kubelet.
1.27-2023 .09.12	1.27.4	1.6.6	1.1.2	Upgrade plugin Amazon VPC CNI untuk menggunakan biner konektor Kubernetes, yang mendapatkan alamat IP Pod dari server API Kubernetes. Permintaan tarik gabungan #100 .

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.27-2023 .08.17	1.27.4	1.6.6	1.1.2	Termasuk tambalan untuk CVE-2023- 3676 , CVE-2023- 3893 , dan CVE-2023- 3955 .
1.27-2023 .08.08	1.27.3	1.6.6	1.1.1	
1.27-2023 .07.11	1.27.3	1.6.6	1.1.1	
1.27-2023 .06.20	1.27.1	1.6.6	1.1.1	Masalah teratasi yang menyebabkan daftar pencarian akhiran DNS salah diisi.
1.27-2023 .06.14	1.27.1	1.6.6	1.1.1	Menambahk an dukungan untuk pemetaan port host di CNI. Permintaan tarik gabungan #93.

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.27-2023 .06.06	1.27.1	1.6.6	1.1.1	Memperbaiki container s-roadmap masalah #2042 , yang menyebabkan node gagal menarik gambar ECR Amazon pribadi.
11.27-202 3.05.18	1.27.1	1.6.6	1.1.1	

Kubernetes versi 1.26

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.26-2025 .05.17	1.26.15	1.7.20	1.2.1	
1.26-2025 .04.14	1.26.15	1.7.20	1.1.3	
1.26-2025 .03.14	1.26.15	1.7.20	1.1.3	Ditingkatkan containerd ke 1.7.20.
1.26-2025 .02.15	1.26.15	1.7.14	1.1.3	
1.26-2025 .01.15	1.26.15	1.7.14	1.1.3	

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.26-2024 .12.11	1.26.15	1.7.14	1.1.3	
1.26-2024 .11.12	1.26.15	1.7.14	1.1.3	
1.26-2024 .10.09	1.26.15	1.7.14	1.1.3	
1.26-2024 .09.10	1.26.15	1.7.14	1.1.3	
1.26-2024 .08.13	1.26.15	1.7.14	1.1.3	
1.26-2024 .07.10	1.26.15	1.7.11	1.1.2	Termasuk tambalan untuk CVE-2024- 5321 .
1.26-2024 .06.17	1.26.15	1.7.11	1.1.2	Ditingkatkan containerd ke 1.7.11.
1.26-2024 .05.14	1.26.15	1.6.28	1.1.2	Ditingkatkan container d ke 1.6.28. Ditingkatkan kubelet ke 1.26.15.

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.26-2024 .04.09	1.26.12	1.6.25	1.1.2	Ditingkatkan containerd ke 1.6.25. Membangun kembali CNI dan csi-proxy menggunakan golang 1.22.1
1.26-2024 .03.13	1.26.12	1.6.18	1.1.2	
1.26-2024 .02.13	1.26.12	1.6.18	1.1.2	
1.26-2024 .01.09	1.26.12	1.6.18	1.1.2	
1.26-2023 .12.12	1.26.10	1.6.18	1.1.2	
1.26-2023 .11.14	1.26.10	1.6.18	1.1.2	Termasuk tambalan untuk CVE-2023-5528 .

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.26-2023.10.19	1.26.9	1.6.18	1.1.2	Ditingkatkan containerd ke 1.6.18. Ditingkatkan kubelet ke 1.26.9. Ditambahkan variabel lingkungan skrip bootstrap baru (SERVICE_I PV4_CIDR dan EXCLUDE SNAT_CIDR S).
1.26-2023.09.12	1.26.7	1.6.6	1.1.2	Upgrade plugin Amazon VPC CNI untuk menggunakan biner konektor Kubernetes, yang mendapatkan alamat IP Pod dari server API Kubernetes. Permintaan tarik gabungan #100 .
1.26-2023.08.17	1.26.7	1.6.6	1.1.2	Termasuk tambalan untuk CVE-2023-3676 , CVE-2023-3893 , dan CVE-2023-3955 .

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.26-2023.08.08	1.26.6	1.6.6	1.1.1	
1.26-2023.07.11	1.26.6	1.6.6	1.1.1	
1.26-2023.06.20	1.26.4	1.6.6	1.1.1	Masalah teratasi yang menyebabkan daftar pencarian akhiran DNS salah diisi.
1.26-2023.06.14	1.26.4	1.6.6	1.1.1	Kubernetes yang ditingkatkan menjadi. 1.26.4 Menambahkan dukungan untuk pemetaan port host di CNI. Permintaan tarik gabungan #93 .

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.26-2023 .05.09	1.26.2	1.6.6	1.1.1	Memperbaiki bug yang menyebabkan masalah konektivitas jaringan #1126 pada pod setelah node restart. Memperkenalkan parameter konfigurasi skrip bootstrap baru (ExcludedS natCIDRs).
1.26-2023 .04.26	1.26.2	1.6.6	1.1.1	
1.26-2023 .04.11	1.26.2	1.6.6	1.1.1	Menambahkan mekanisme pemulihan untuk kubelet dan kube-proxy pada kerusakan layanan.
1.26-2023 .03.24	1.26.2	1.6.6	1.1.1	

Windows Server 2019 Full AMI yang dioptimalkan oleh Amazon EKS

Tabel berikut mencantumkan versi saat ini dan sebelumnya dari Amazon EKS yang dioptimalkan Windows Server 2019 Full AMI.

Example

Kubernetes versi 1.33

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.33-2025 .06.13	1.33.1	1.7.27	1.2.1	
1.33-2025 .05.17	1.33.1	1.7.27	1.2.1	

Kubernetes versi 1.32

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.32-2025 .06.13	1.32.5	1.7.27	1.2.1	Ditingkatkan containerd ke1.7.27.
1.32-2025 .05.17	1.32.5	1.7.20	1.2.1	
1.32-2025 .04.14	1.32.1	1.7.20	1.1.3	
1.32-2025 .03.14	1.32.1	1.7.20	1.1.3	
1.32-2025 .02.18	1.32.1	1.7.20	1.1.3	
1.32-2025 .01.15	1.32.0	1.7.20	1.1.3	

Kubernetes versi 1.31

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.31-2025 .06.13	1.31.9	1.7.27	1.2.1	Ditingkatkan containerd ke1.7.27.
1.31-2025 .05.17	1.31.9	1.7.20	1.2.1	
1.31-2025 .04.14	1.31.5	1.7.20	1.1.3	
1.31-2025 .03.14	1.31.5	1.7.20	1.1.3	
1.31-2025 .02.15	1.31.5	1.7.20	1.1.3	
1.31-2025 .01.15	1.31.4	1.7.20	1.1.3	
1.31-2025 .01.01	1.31.4	1.7.20	1.1.3	Termasuk tambalan untukCVE-2024- 9042 .
1.31-2024 .12.13	1.31.3	1.7.20	1.1.3	
1.31-2024 .11.12	1.31.1	1.7.20	1.1.3	
1.31-2024 .10.08	1.31.1	1.7.20	1.1.3	
1.31-2024 .10.01	1.31.1	1.7.20	1.1.3	

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.31-2024 .09.10	1.31.0	1.7.20	1.1.3	

Kubernetes versi 1.30

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.30-2025 .06.13	1.30.13	1.7.27	1.2.1	Ditingkatkan containerd ke1.7.27.
1.30-2025 .05.17	1.30.13	1.7.20	1.2.1	
1.30-2025 .04.14	1.30.9	1.7.20	1.1.3	
1.30-2025 .03.14	1.30.9	1.7.20	1.1.3	Ditingkatkan containerd ke1.7.20.
1.30-2025 .02.15	1.30.9	1.7.14	1.1.3	
1.30-2025 .01.15	1.30.8	1.7.14	1.1.3	
1.30-2025 .01.01	1.30.8	1.7.14	1.1.3	Termasuk tambalan untukCVE-2024- 9042 .
1.30-2024 .12.11	1.30.7	1.7.14	1.1.3	

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.30-2024.11.12	1.30.4	1.7.14	1.1.3	
1.30-2024.10.08	1.30.4	1.7.14	1.1.3	
1.30-2024.09.10	1.30.2	1.7.14	1.1.3	
1.30-2024.08.13	1.30.2	1.7.14	1.1.3	
1.30-2024.07.10	1.30.2	1.7.14	1.1.2	Termasuk tambalan untuk CVE-2024-5321 .
1.30-2024.06.17	1.30.0	1.7.14	1.1.2	Ditingkatkan containerd ke 1.7.14.
1.30-2024.05.15	1.30.0	1.6.28	1.1.2	

Kubernetes versi 1.29

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.29-2025.06.13	1.29.15	1.7.27	1.2.1	Ditingkatkan containerd ke 1.7.27.
1.29-2025.05.17	1.29.15	1.7.20	1.2.1	

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.29-2025 .04.14	1.29.13	1.7.20	1.1.3	
1.29-2025 .03.14	1.29.13	1.7.20	1.1.3	Ditingkatkan containerd ke1.7.20.
1.29-2025 .02.15	1.29.13	1.7.14	1.1.3	
1.29-2025 .01.15	1.29.12	1.7.14	1.1.3	
1.29-2025 .01.01	1.29.12	1.7.14	1.1.3	Termasuk tambalan untukCVE-2024- 9042 .
1.29-2024 .12.11	1.29.10	1.7.14	1.1.3	
1.29-2024 .11.12	1.29.8	1.7.14	1.1.3	
1.29-2024 .10.08	1.29.8	1.7.14	1.1.3	
1.29-2024 .09.10	1.29.6	1.7.14	1.1.3	
1.29-2024 .08.13	1.29.6	1.7.14	1.1.3	
1.29-2024 .07.10	1.29.6	1.7.11	1.1.2	Termasuk tambalan untukCVE-2024- 5321 .

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.29-2024.06.17	1.29.3	1.7.11	1.1.2	
1.29-2024.05.15	1.29.3	1.7.11	1.1.2	Ditingkatkan containerd ke 1.7.11. Ditingkatkan kubelet ke 1.29.3.
1.29-2024.04.09	1.29.0	1.6.28	1.1.2	Ditingkatkan containerd ke 1.6.28. Membangun kembali CNI dan csi-proxy menggunakan golang 1.22.1
1.29-2024.03.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.06	1.29.0	1.6.25	1.1.2	Memperbaiki bug di mana gambar jeda salah dihapus oleh proses pengumpulan kubelet sampah.

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.29-2024 .01.09	1.29.0	1.6.18	1.1.2	

Kubernetes versi 1.28

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.28-2025 .06.13	1.28.15	1.7.27	1.2.1	Ditingkatkan containerd ke1.7.27.
1.28-2025 .05.17	1.28.15	1.7.20	1.2.1	
1.28-2025 .04.14	1.28.15	1.7.20	1.1.3	
1.28-2025 .03.14	1.28.15	1.7.20	1.1.3	Ditingkatkan containerd ke1.7.20.
1.28-2025 .02.15	1.28.15	1.7.14	1.1.3	
1.28-2025 -01-15	1.28.15	1.7.14	1.1.3	
1.28-2025 -01-01	1.28.15	1.7.14	1.1.3	Termasuk tambalan untukCVE-2024- 9042 .
1.28-2024 .12.11	1.28.15	1.7.14	1.1.3	

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.28-2024 .11.12	1.28.13	1.7.14	1.1.3	
1.28-2024 .10.08	1.28.13	1.7.14	1.1.3	
1.28-2024 .09.10	1.28.11	1.7.14	1.1.3	
1.28-2024 .08.13	1.28.11	1.7.14	1.1.3	
1.28-2024 .07.10	1.28.11	1.7.11	1.1.2	Termasuk tambalan untuk CVE-2024- 5321 .
1.28-2024 .06.17	1.28.8	1.7.11	1.1.2	Ditingkatkan containerd ke 1.7.11.
1.28-2024 .05.14	1.28.8	1.6.28	1.1.2	Ditingkatkan container d ke 1.6.28. Ditingkatkan kubelet ke 1.28.8.

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.28-2024 .04.09	1.28.5	1.6.25	1.1.2	Ditingkatkan containerd ke 1.6.25. Membangun kembali CNI dan csi-proxy menggunakan golang 1.22.1
1.28-2024 .03.13	1.28.5	1.6.18	1.1.2	
1.28-2024 .02.13	1.28.5	1.6.18	1.1.2	
1.28-2024 .01.09	1.28.5	1.6.18	1.1.2	
1.28-2023 .12.12	1.28.3	1.6.18	1.1.2	
1.28-2023 .11.14	1.28.3	1.6.18	1.1.2	Termasuk tambalan untuk CVE-2023-5528 .

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.28-2023 .10.19	1.28.2	1.6.18	1.1.2	Ditingkatkan containerd ke 1.6.18. Ditambahkan variabel lingkungan skrip bootstrap baru (SERVICE_IPV4_CIDR dan EXCLUDE_SNAT_CIDR S).
1.28-2023 -09.27	1.28.2	1.6.6	1.1.2	Memperbaiki penasihat keamanan di kubelet.
1.28-2023 .09.12	1.28.1	1.6.6	1.1.2	

Kubernetes versi 1.27

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.27-2025 .06.13	1.27.16	1.7.27	1.2.1	Ditingkatkan containerd ke 1.7.27.
1.27-2025 .05.17	1.27.16	1.7.20	1.2.1	
1.27-2025 .04.14	1.27.16	1.7.20	1.1.3	

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.27-2025 .03.14	1.27.16	1.7.20	1.1.3	Ditingkatkan containerd ke1.7.20.
1.27-2025 .02.15	1.27.16	1.7.14	1.1.3	
1.27-2025 .01.15	1.27.16	1.7.14	1.1.3	
1.27-2025 .01.01	1.27.16	1.7.14	1.1.3	Termasuk tambalan untukCVE-2024- 9042 .
1.27-2024 .12.11	1.27.16	1.7.14	1.1.3	
1.27-2024 .11.12	1.27.16	1.7.14	1.1.3	
1.27-2024 .10.08	1.27.16	1.7.14	1.1.3	
1.27-2024 .09.10	1.27.15	1.7.14	1.1.3	
1.27-2024 .08.13	1.27.15	1.7.14	1.1.3	
1.27-2024 .07.10	1.27.15	1.7.11	1.1.2	Termasuk tambalan untukCVE-2024- 5321 .

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.27-2024.06.17	1.27.12	1.7.11	1.1.2	Ditingkatkan containerd ke 1.7.11.
1.27-2024.05.14	1.27.12	1.6.28	1.1.2	Ditingkatkan containerd ke 1.6.28. Ditingkatkan kubelet ke 1.27.12.
1.27-2024.04.09	1.27.9	1.6.25	1.1.2	Ditingkatkan containerd ke 1.6.25. Membangun kembali CNI dan csi-proxy menggunakan golang 1.22.1
1.27-2024.03.13	1.27.9	1.6.18	1.1.2	
1.27-2024.02.13	1.27.9	1.6.18	1.1.2	
1.27-2024.01.09	1.27.9	1.6.18	1.1.2	
1.27-2023.12.12	1.27.7	1.6.18	1.1.2	

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.27-2023 .11.14	1.27.7	1.6.18	1.1.2	Termasuk tambalan untuk CVE-2023-5528 .
1.27-2023 .10.19	1.27.6	1.6.18	1.1.2	Ditingkatkan containerd ke 1.6.18. Ditambahkan variabel lingkungan skrip bootstrap baru (SERVICE_IPV4_CIDR dan EXCLUDE_SNAT_CIDR S).
1.27-2023 -09.27	1.27.6	1.6.6	1.1.2	Memperbaiki penasihat keamanan di kubelet.
1.27-2023 .09.12	1.27.4	1.6.6	1.1.2	Upgrade plugin Amazon VPC CNI untuk menggunakan biner konektor Kubernetes, yang mendapatkan alamat IP Pod dari server API Kubernetes. Permintaan tarik gabungan #100 .

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.27-2023 .08.17	1.27.4	1.6.6	1.1.2	Termasuk tambalan untuk CVE-2023- 3676 , CVE-2023- 3893 , dan CVE-2023- 3955 .
1.27-2023 .08.08	1.27.3	1.6.6	1.1.1	
1.27-2023 .07.11	1.27.3	1.6.6	1.1.1	
1.27-2023 .06.20	1.27.1	1.6.6	1.1.1	Masalah teratasi yang menyebabkan daftar pencarian akhiran DNS salah diisi.
1.27-2023 .06.14	1.27.1	1.6.6	1.1.1	Menambahk an dukungan untuk pemetaan port host di CNI. Permintaan tarik gabungan #93.

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.27-2023 .06.06	1.27.1	1.6.6	1.1.1	Memperbaiki container s-roadmap masalah #2042 , yang menyebabkan node gagal menarik gambar ECR Amazon pribadi.
1.27-2023 .05.17	1.27.1	1.6.6	1.1.1	

Kubernetes versi 1.26

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.26-2025 .05.17	1.26.15	1.7.20	1.2.1	
1.26-2025 .04.14	1.26.15	1.7.20	1.1.3	
1.26-2025 .03.14	1.26.15	1.7.20	1.1.3	Ditingkatkan containerd ke 1.7.20.
1.26-2025 .02.15	1.26.15	1.7.14	1.1.3	
1.26-2025 .01.15	1.26.15	1.7.14	1.1.3	

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.26-2024 .12.11	1.26.15	1.7.14	1.1.3	
1.26-2024 .11.12	1.26.15	1.7.14	1.1.3	
1.26-2024 .10.08	1.26.15	1.7.14	1.1.3	
1.26-2024 .09.10	1.26.15	1.7.14	1.1.3	
1.26-2024 .08.13	1.26.15	1.7.14	1.1.3	
1.26-2024 .07.10	1.26.15	1.7.11	1.1.2	Termasuk tambalan untuk CVE-2024- 5321 .
1.26-2024 .06.17	1.26.15	1.7.11	1.1.2	Ditingkatkan containerd ke 1.7.11.
1.26-2024 .05.14	1.26.15	1.6.28	1.1.2	Ditingkatkan container d ke 1.6.28. Ditingkatkan kubelet ke 1.26.15.

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.26-2024 .04.09	1.26.12	1.6.25	1.1.2	Ditingkatkan containerd ke 1.6.25. Membangun kembali CNI dan csi-proxy menggunakan an. golang 1.22.1
1.26-2024 .03.13	1.26.12	1.6.18	1.1.2	
1.26-2024 .02.13	1.26.12	1.6.18	1.1.2	
1.26-2024 .01.09	1.26.12	1.6.18	1.1.2	
1.26-2023 .12.12	1.26.10	1.6.18	1.1.2	
1.26-2023 .11.14	1.26.10	1.6.18	1.1.2	Termasuk tambalan untuk CVE-2023-5528 .

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.26-2023.10.19	1.26.9	1.6.18	1.1.2	Ditingkatkan containerd ke 1.6.18. Ditingkatkan kubelet ke 1.26.9. Ditambahkan variabel lingkungan skrip bootstrap baru (SERVICE_IP, PV4_CIDR dan EXCLUDE_SNAT_CIDR S).
1.26-2023.09.12	1.26.7	1.6.6	1.1.2	Upgrade plugin Amazon VPC CNI untuk menggunakan biner konektor Kubernetes, yang mendapatkan alamat IP Pod dari server API Kubernetes. Permintaan tarik gabungan #100 .
1.26-2023.08.17	1.26.7	1.6.6	1.1.2	Termasuk tambalan untuk CVE-2023-3676 , CVE-2023-3893 , dan CVE-2023-3955 .

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.26-2023.08.08	1.26.6	1.6.6	1.1.1	
1.26-2023.07.11	1.26.6	1.6.6	1.1.1	
1.26-2023.06.20	1.26.4	1.6.6	1.1.1	Masalah teratasi yang menyebabkan daftar pencarian akhiran DNS salah diisi.
1.26-2023.06.14	1.26.4	1.6.6	1.1.1	Kubernetes yang ditingkatkan menjadi. 1.26.4 Menambahkan dukungan untuk pemetaan port host di CNI. Permintaan tarik gabungan #93 .

Versi AMI	versi kubelet	versi containerd	versi csi-proxy	Catatan rilis
1.26-2023 .05.09	1.26.2	1.6.6	1.1.1	Memperbaiki bug yang menyebabkan masalah konektivitas jaringan #1126 pada pod setelah node restart. Memperkenalkan parameter konfigurasi skrip bootstrap baru (ExcludedS natCIDRs).
1.26-2023 .04.26	1.26.2	1.6.6	1.1.1	
1.26-2023 .04.11	1.26.2	1.6.6	1.1.1	Menambahkan mekanisme pemulihan untuk kubelet dan kube-proxy pada kerusakan layanan.
1.26-2023 .03.24	1.26.2	1.6.6	1.1.1	

Ambil Microsoft Windows AMI yang direkomendasikan IDs

Saat menerapkan node, Anda dapat menentukan ID untuk Amazon Machine Image (AMI) yang dioptimalkan Amazon EKS yang telah dibuat sebelumnya. Untuk mengambil ID AMI yang sesuai dengan konfigurasi yang Anda inginkan, kueri AWS Systems Manager Parameter Store API.

Menggunakan API ini menghilangkan kebutuhan untuk secara manual mencari AMI Amazon EKS yang dioptimalkan IDs. Untuk informasi selengkapnya, lihat [GetParameter](#). [Prinsip IAM](#) yang Anda gunakan harus memiliki izin `ssm:GetParameter` IAM untuk mengambil metadata AMI Amazon EKS yang dioptimalkan.

Anda dapat mengambil ID gambar dari Amazon EKS terbaru yang dioptimalkan Windows AMI yang dioptimalkan dengan perintah berikut, yang menggunakan `image_id` sub-parameter. Buat modifikasi berikut pada perintah sesuai kebutuhan dan kemudian jalankan perintah yang dimodifikasi:

- Ganti *release* dengan salah satu opsi berikut.
 - Gunakan *2022* untuk Windows Server 2022.
 - Gunakan *2019* untuk Windows Server 2019.
- Ganti *installation-option* dengan salah satu opsi berikut. Untuk informasi selengkapnya, lihat [Apa itu opsi instalasi inti server di Windows Server](#).
 - Gunakan *Core* untuk instalasi minimal dengan permukaan serangan yang lebih kecil.
 - Gunakan *Full* untuk menyertakan pengalaman desktop Windows.
- Ganti *kubernetes-version* dengan [versi Amazon EKS](#) yang didukung.
- Ganti *region-code* dengan [AWS Wilayah yang didukung Amazon EKS](#) yang Anda inginkan ID AMI.

```
aws ssm get-parameter --name /aws/service/ami-windows-latest/Windows_Server-release-English-installation-option-EKS_Optimized-kubernetes-version/image_id \  
--region region-code --query "Parameter.Value" --output text
```

Berikut adalah contoh perintah setelah penggantian placeholder dibuat.

```
aws ssm get-parameter --name /aws/service/ami-windows-latest/Windows_Server-2022-English-Core-EKS_Optimized-k8s-n-2/image_id \  
--region us-west-2 --query "Parameter.Value" --output text
```

Contoh output adalah sebagai berikut.

```
ami-1234567890abcdef0
```

Membangun AMI Windows kustom dengan Image Builder

Anda dapat menggunakan EC2 Image Builder untuk membuat Windows Amazon EKS yang dioptimalkan khusus AMIs dengan salah satu opsi berikut:

- [Menggunakan Amazon EKS dioptimalkan Windows AMI sebagai basis](#)
- [Menggunakan komponen build yang dikelola Amazon](#)

Dengan kedua metode tersebut, Anda harus membuat resep Image Builder Anda sendiri. Untuk informasi selengkapnya, lihat [Membuat versi baru resep gambar](#) di Panduan Pengguna Image Builder.

Important

Komponen yang dikelola Amazon berikut untuk eks menyertakan tambalan untuk CVE-2024-5321

- 1.26.4 dan lebih tinggi
- 1.27.2 dan lebih tinggi
- 1.28.2 dan lebih tinggi
- 1.29.2 dan lebih tinggi
- 1.30.1 dan lebih tinggi

Menggunakan Amazon EKS dioptimalkan Windows AMI sebagai basis

Opsi ini adalah cara yang disarankan untuk membangun Windows kustom Anda AMIs. Windows Amazon EKS yang dioptimalkan yang AMIs kami sediakan lebih sering diperbarui daripada komponen build yang dikelola Amazon.

1. Mulai resep Image Builder baru.
 - a. Buka konsol EC2 Image Builder di <https://console.aws.amazon.com/imagebuilder>.
 - b. Di panel navigasi kiri, pilih Resep gambar.
 - c. Pilih Buat resep gambar.
2. Di bagian Rincian resep, masukkan Nama dan Versi.
3. Tentukan ID Amazon EKS Windows AMI yang dioptimalkan di bagian Gambar dasar.

- a. Pilih Masukkan ID AMI khusus.
 - b. Ambil ID AMI untuk versi OS Windows yang Anda butuhkan. Untuk informasi selengkapnya, lihat [the section called “Dapatkan yang terbaru IDs”](#).
 - c. Masukkan ID AMI kustom. Jika ID AMI tidak ditemukan, pastikan AWS Region untuk ID AMI cocok dengan AWS Region yang ditampilkan di kanan atas konsol Anda.
4. (Opsional) Untuk mendapatkan pembaruan keamanan terbaru, tambahkan update-windows komponen di bagian Build components -.
- a. Dari daftar dropdown di sebelah kanan kotak pencarian Cari komponen berdasarkan nama, pilih dikelola Amazon.
 - b. Di kotak Cari komponen dengan nama pencarian, masukkan update-windows.
 - c. Pilih kotak centang hasil **update-windows** pencarian. Komponen ini mencakup patch Windows terbaru untuk sistem operasi.
5. Lengkapi masukan resep gambar yang tersisa dengan konfigurasi yang Anda butuhkan. Untuk informasi selengkapnya, lihat [Membuat versi resep gambar baru \(konsol\)](#) di Panduan Pengguna Image Builder.
6. Pilih Buat resep.
7. Gunakan resep gambar baru di pipeline gambar baru atau yang sudah ada. Setelah pipeline gambar Anda berhasil berjalan, AMI kustom Anda akan terdaftar sebagai gambar keluaran dan siap digunakan. Untuk informasi selengkapnya, lihat [Membuat pipeline gambar menggunakan wizard konsol EC2 Image Builder](#).

Menggunakan komponen build yang dikelola Amazon

Saat menggunakan AMI Windows Amazon EKS yang dioptimalkan sebagai basis tidak layak, Anda dapat menggunakan komponen build yang dikelola Amazon sebagai gantinya. Opsi ini mungkin tertinggal dari versi Kubernetes terbaru yang didukung.

1. Mulai resep Image Builder baru.
 - a. Buka konsol EC2 Image Builder di <https://console.aws.amazon.com/imagebuilder>.
 - b. Di panel navigasi kiri, pilih Resep gambar.
 - c. Pilih Buat resep gambar.
2. Di bagian Rincian resep, masukkan Nama dan Versi.
3. Tentukan opsi mana yang akan Anda gunakan untuk membuat AMI kustom Anda di bagian Gambar dasar:

- Pilih gambar terkelola - Pilih Windows untuk Sistem Operasi Gambar (OS) Anda. Kemudian pilih salah satu opsi berikut untuk Asal gambar.
 - Mulai cepat (dikelola Amazon) - Di dropdown nama Gambar, pilih versi Windows Server yang didukung Amazon EKS. Untuk informasi selengkapnya, lihat [the section called “Windows”](#).
 - Gambar yang dimiliki oleh saya - Untuk nama Gambar, pilih ARN gambar Anda sendiri dengan lisensi Anda sendiri. Gambar yang Anda berikan belum dapat menginstal komponen Amazon EKS.
 - Masukkan ID AMI khusus — Untuk ID AMI, masukkan ID untuk AMI Anda dengan lisensi Anda sendiri. Gambar yang Anda berikan belum dapat menginstal komponen Amazon EKS.
4. Di bagian Build components - Windows, lakukan hal berikut:
- a. Dari daftar dropdown di sebelah kanan kotak pencarian Cari komponen berdasarkan nama, pilih dikelola Amazon.
 - b. Di kotak Cari komponen dengan nama pencarian, masukkan eks.
 - c. Pilih kotak centang hasil **eks-optimized-ami-windows** pencarian, meskipun hasil yang dikembalikan mungkin bukan versi yang Anda inginkan.
 - d. Di kotak Cari komponen dengan nama pencarian, masukkan update-windows.
 - e. Pilih kotak centang hasil pencarian pembaruan-jendela. Komponen ini mencakup patch Windows terbaru untuk sistem operasi.
5. Di bagian Komponen yang dipilih, lakukan hal berikut:
- a. Pilih opsi Versioning untuk **eks-optimized-ami-windows**
 - b. Pilih Tentukan versi komponen.
 - c. Di bidang Component Version, enter **version.x**, ganti **version** dengan versi Kubernetes yang didukung. Memasukkan bagian **x** untuk nomor versi menunjukkan untuk menggunakan versi komponen terbaru yang juga sejajar dengan bagian versi yang Anda tentukan secara eksplisit. Perhatikan output konsol karena akan memberi tahu Anda apakah versi yang Anda inginkan tersedia sebagai komponen terkelola. Perlu diingat bahwa versi Kubernetes terbaru mungkin tidak tersedia untuk komponen build. Untuk informasi selengkapnya tentang versi yang tersedia, lihat [the section called “Mengambil informasi tentang versi eks-optimized-ami-windows komponen”](#).
6. Lengkapi masukan resep gambar yang tersisa dengan konfigurasi yang Anda butuhkan. Untuk informasi selengkapnya, lihat [Membuat versi resep gambar baru \(konsol\)](#) di Panduan Pengguna Image Builder.
7. Pilih Buat resep.

- Gunakan resep gambar baru di pipeline gambar baru atau yang sudah ada. Setelah pipeline gambar Anda berhasil berjalan, AMI kustom Anda akan terdaftar sebagai gambar keluaran dan siap digunakan. Untuk informasi selengkapnya, lihat [Membuat pipeline gambar menggunakan wizard konsol EC2 Image Builder](#).

Mengambil informasi tentang versi **eks-optimized-ami-windows** komponen

Anda dapat mengambil informasi spesifik mengenai apa yang diinstal dengan masing-masing komponen. Misalnya, Anda dapat memverifikasi kubelet versi apa yang diinstal. Komponen melalui pengujian fungsional pada Amazon EKS mendukung versi sistem operasi Windows. Untuk informasi selengkapnya, lihat [the section called “Kalender rilis”](#). Versi OS Windows lainnya yang tidak terdaftar sebagai didukung atau telah mencapai akhir dukungan mungkin tidak kompatibel dengan komponen.

- Buka konsol EC2 Image Builder di <https://console.aws.amazon.com/imagebuilder>.
- Di panel navigasi kiri, pilih Komponen.
- Dari daftar dropdown di sebelah kanan kotak pencarian Cari komponen berdasarkan nama, ubah Dimiliki oleh saya menjadi Mulai cepat (dikelola Amazon).
- Dalam kotak Temukan komponen berdasarkan nama, masukkan eks.
- (Opsional) Jika Anda menggunakan versi terbaru, urutkan kolom Versi dalam urutan menurun dengan memilihnya dua kali.
- Pilih **eks-optimized-ami-window**stautan dengan versi yang diinginkan.

Deskripsi di halaman yang dihasilkan menunjukkan informasi spesifik.

Aktifkan perbaikan otomatis node dan selidiki masalah kesehatan node

Kesehatan node mengacu pada status operasional dan kemampuan node untuk menjalankan beban kerja secara efektif. Node yang sehat mempertahankan konektivitas yang diharapkan, memiliki sumber daya yang cukup, dan dapat berhasil menjalankan Pod tanpa gangguan. Untuk informasi tentang mendapatkan detail tentang node Anda, lihat [the section called “Lihat kesehatan simpul”](#) dan [the section called “Dapatkan log simpul”](#).

Untuk membantu menjaga node yang sehat, Amazon EKS menawarkan agen pemantauan node dan perbaikan otomatis node.

 Important

Agan pemantauan node dan perbaikan otomatis node hanya tersedia di Linux. Fitur ini tidak tersedia di Windows.

Agan pemantauan node

Agan pemantauan node secara otomatis membaca log node untuk mendeteksi masalah kesehatan tertentu. Ini mem-parsing melalui log node untuk mendeteksi kegagalan dan memunculkan berbagai informasi status tentang node pekerja. Dedicated `NodeCondition` diterapkan pada node pekerja untuk setiap kategori masalah yang terdeteksi, seperti masalah penyimpanan dan jaringan. Deskripsi masalah kesehatan yang terdeteksi tersedia di dasbor observabilitas. Untuk informasi selengkapnya, lihat [the section called “Masalah kesehatan simpul”](#).

Agan pemantauan node disertakan sebagai kemampuan untuk semua kluster Mode Otomatis Amazon EKS. Untuk jenis cluster lainnya, Anda dapat menambahkan agan pemantauan sebagai add-on Amazon EKS. Untuk informasi selengkapnya, lihat [the section called “Buat add-on”](#).

Perbaikan otomatis node

Node auto repair adalah fitur tambahan yang terus memantau kesehatan node, secara otomatis bereaksi terhadap masalah yang terdeteksi dan mengganti node bila memungkinkan. Ini membantu ketersediaan cluster secara keseluruhan dengan intervensi manual minimal. Jika pemeriksaan kesehatan gagal, node akan ditutup secara otomatis sehingga tidak ada Pod baru yang dijadwalkan pada node.

Dengan sendirinya, perbaikan otomatis node dapat bereaksi terhadap Ready kondisi kubelet dan objek node apa pun yang dihapus secara manual. Ketika dipasangkan dengan agan pemantauan node, perbaikan otomatis node dapat bereaksi terhadap lebih banyak kondisi yang tidak akan terdeteksi sebaliknya. Kondisi tambahan ini termasuk `KernelReady`, `NetworkingReady`, dan `StorageReady`.

Pemulihan node otomatis ini secara otomatis mengatasi masalah node intermiten seperti kegagalan untuk bergabung dengan cluster, kubelet yang tidak responsif, dan peningkatan kesalahan akselerator (perangkat). Keandalan yang ditingkatkan membantu mengurangi waktu henti aplikasi dan meningkatkan operasi klaster. Node auto repair tidak dapat menangani masalah tertentu yang dilaporkan seperti `DiskPressure`, `MemoryPressure`, dan `PIDPressure`. Amazon EKS menunggu

10 menit sebelum bertindak `AcceleratedHardwareReadyNodeConditions`, dan 30 menit untuk semua kondisi lainnya.

Grup node terkelola juga akan secara otomatis menonaktifkan perbaikan node untuk alasan keamanan di bawah dua skenario. Setiap operasi perbaikan yang sebelumnya sedang berlangsung akan berlanjut untuk kedua situasi.

- Jika pergeseran zona untuk klaster Anda telah dipicu melalui Application Recovery Controller (ARC), semua operasi perbaikan berikutnya dihentikan.
- Jika grup node Anda memiliki lebih dari lima node dan lebih dari 20% node dalam grup node Anda berada dalam keadaan tidak sehat, operasi perbaikan dihentikan.

Anda dapat mengaktifkan perbaikan otomatis node saat membuat atau mengedit grup node terkelola.

- Saat menggunakan konsol Amazon EKS, aktifkan kotak centang Aktifkan perbaikan otomatis node untuk grup node terkelola. Untuk informasi selengkapnya, lihat [the section called “Buat”](#).
- Saat menggunakan AWS CLI, tambahkan `--node-repair-config enabled=true` ke perintah [eks create nodegroup](#) or [eks update-nodegroup-config](#).
- Untuk contoh `eksctl ClusterConfig` yang menggunakan grup node terkelola dengan perbaikan otomatis node, lihat [44-node-repair.yaml](#) aktif. GitHub

Masalah kesehatan simpul

Tabel berikut menjelaskan masalah kesehatan simpul yang dapat dideteksi oleh agen pemantauan node. Ada dua jenis masalah:

- Kondisi — Masalah terminal yang menjamin tindakan remediasi seperti penggantian instance atau reboot. Ketika perbaikan otomatis diaktifkan, Amazon EKS akan melakukan tindakan perbaikan, baik sebagai penggantian node atau reboot. Untuk informasi selengkapnya, lihat [the section called “Kondisi simpul”](#).
- Event — Masalah sementara atau konfigurasi node sub-optimal. Tidak ada tindakan perbaikan otomatis yang akan terjadi. Untuk informasi selengkapnya, lihat [the section called “Peristiwa simpul”](#).

Masalah kesehatan simpul kernel

Nama	Kepelikan	Deskripsi
ForkFailedOutOfPID	Ketentuan	Panggilan fork atau exec gagal karena sistem kehabisan proses IDs atau memori, yang mungkin disebabkan oleh proses zombie atau kelelahan memori fisik.
AppBlocked	Peristiwa	Tugas telah diblokir untuk jangka waktu yang lama dari penjadwalan, biasanya disebabkan oleh diblokir pada input atau output.
AppCrash	Peristiwa	Aplikasi pada node telah crash.
ApproachingKernelPidMax	Peristiwa	Jumlah proses mendekati jumlah maksimum PIDs yang tersedia per pengaturan kernel.pid_max saat ini, setelah itu tidak ada lagi proses yang dapat diluncurkan.
ApproachingMaxOpenFiles	Peristiwa	Jumlah file yang terbuka mendekati jumlah maksimum file terbuka yang mungkin diberikan pengaturan kernel saat ini, setelah itu membuka file baru akan gagal.
ConntrackExceededKernel	Peristiwa	Pelacakan koneksi melebihi maksimum untuk kernel dan

Nama	Kepelikan	Deskripsi
		koneksi baru tidak dapat dibuat, yang dapat mengakibatkan hilangnya paket.
ExcessiveZombieProcesses	Peristiwa	Proses yang tidak dapat sepenuhnya direklamasi terakumulasi dalam jumlah besar, yang menunjukkan masalah aplikasi dan dapat menyebabkan mencapai batas proses sistem.
KernelBug	Peristiwa	Bug kernel terdeteksi dan dilaporkan oleh kernel Linux itu sendiri, meskipun ini kadang-kadang disebabkan oleh node dengan CPU tinggi atau penggunaan memori yang menyebabkan pemrosesan peristiwa tertunda.
LargeEnvironment	Peristiwa	Jumlah variabel lingkungan untuk proses ini lebih besar dari yang diharapkan, berpotensi disebabkan oleh banyak layanan dengan enableServiceLinks set ke true, yang dapat menyebabkan masalah kinerja.

Nama	Kepelikan	Deskripsi
RapidCron	Peristiwa	Pekerjaan cron berjalan lebih cepat daripada setiap lima menit pada node ini, yang dapat memengaruhi kinerja jika pekerjaan tersebut menghabiskan sumber daya yang signifikan.
SoftLockup	Peristiwa	CPU terhenti untuk jangka waktu tertentu.

Masalah kesehatan simpul jaringan

Nama	Kepelikan	Deskripsi
InterfaceNotRunning	Ketentuan	Antarmuka ini tampaknya tidak berjalan atau ada masalah jaringan.
InterfaceNotUp	Ketentuan	Antarmuka ini tampaknya tidak aktif atau ada masalah jaringan.
IPAMDNotSiap	Ketentuan	IPAMD gagal terhubung ke server API.
IPAMDNotBerlari	Ketentuan	aws-k8s-agent Prosesnya tidak ditemukan berjalan.
MissingLoopbackInterface	Ketentuan	Antarmuka loopback hilang dari instance ini, menyebabkan kegagalan layanan tergantung pada konektivitas lokal.

Nama	Kepelikan	Deskripsi
BandwidthInExceeded	Peristiwa	Paket telah diantrian atau dijatuhkan karena bandwidth agregat masuk melebihi maksimum untuk instance.
BandwidthOutExceeded	Peristiwa	Paket telah diantrian atau dijatuhkan karena bandwidth agregat keluar melebihi maksimum untuk instance.
ConntrackExceeded	Peristiwa	Pelacakan koneksi melebihi maksimum untuk instance dan koneksi baru tidak dapat dibuat, yang dapat mengakibatkan hilangnya paket.
IPAMDNSNoIPs	Peristiwa	IPAM-D kehabisan alamat IP.
IPAMDNSRepeatedlyMulai ulang	Peristiwa	Beberapa restart dalam layanan IPAMDNS telah terjadi.
KubeProxyNotReady	Peristiwa	Kube-proxy gagal menonton atau mencantumkan sumber daya.
LinkLocalExceeded	Peristiwa	Paket dijatuhkan karena PPS lalu lintas ke layanan proxy lokal melebihi maksimum antarmuka jaringan.
MissingDefaultRoutes	Peristiwa	Ada aturan rute default yang hilang.
HilangIPRules, Hilang IPRoutes	Peristiwa	Ada aturan rute yang hilang untuk Pod berikut IPs dari tabel rute.

Nama	Kepelikan	Deskripsi
NetworkSysctl	Peristiwa	Pengaturan sysctl jaringan node ini berpotensi salah.
PortConflict	Peristiwa	Jika sebuah Pod menggunakan HostPort, ia dapat menulis aturan iptables yang mengganti port host yang sudah terikat, yang berpotensi mencegah akses server API. kubelet
PPSExceeded	Peristiwa	Paket telah diantrian atau dijatuhkan karena PPS dua arah melebihi maksimum untuk instance.
UnexpectedRejectRule	Peristiwa	Sebuah tak terduga REJECT atau DROP aturan ditemukan di iptables, berpotensi memblokir lalu lintas yang diharapkan.

Masalah kesehatan simpul neuron

Nama	Kepelikan	Deskripsi
Neuron DMAError	Ketentuan	Mesin DMA mengalami kesalahan yang tidak dapat dipulihkan.
HBMUncorrectableKesalahan Neuron	Ketentuan	HBM mengalami kesalahan yang tidak dapat diperbaiki dan menghasilkan hasil yang salah.

Nama	Kepelikan	Deskripsi
NCUncorrectableKesalahan Neuron	Ketentuan	Kesalahan memori Neuron Core yang tidak dapat diperbaiki terdeteksi.
SRAMUncorrectableKesalahan Neuron	Ketentuan	SRAM on-chip mengalami kesalahan paritas dan menghasilkan hasil yang salah.

Masalah kesehatan node NVIDIA

Jika perbaikan otomatis diaktifkan, tindakan perbaikan yang tercantum mulai 10 menit setelah masalah terdeteksi. Untuk informasi selengkapnya tentang kesalahan XID, lihat Kesalahan [Xid](#) di Deployment GPU NVIDIA dan Dokumentasi Manajemen. Untuk informasi selengkapnya tentang pesan XID individual, lihat [Memahami Pesan Xid di Dokumentasi](#) Penerapan dan Manajemen GPU NVIDIA.

Nama	Kepelikan	Deskripsi	Tindakan perbaikan
NvidiaDoubleBitError	Ketentuan	Kesalahan bit ganda dihasilkan oleh driver GPU.	Ganti
NVLinkKesalahan Nvidia	Ketentuan	NVLink kesalahan dilaporkan oleh driver GPU.	Ganti
XID13Kesalahan Nvidia	Ketentuan	Ada pengecualian mesin grafis.	Mulai ulang
XID31Kesalahan Nvidia	Ketentuan	Ada dugaan masalah perangkat keras.	Mulai ulang

Nama	Kepelikan	Deskripsi	Tindakan perbaikan
XID48Kesalahan Nvidia	Ketentuan	Kesalahan ECC bit ganda dilaporkan oleh pengemudi.	Mulai ulang
XID63Kesalahan Nvidia	Ketentuan	Ada halaman pensiun atau pemetaan ulang baris.	Mulai ulang
XID64Kesalahan Nvidia	Ketentuan	Ada kegagalan mencoba untuk menghentikan halaman atau melakukan pemetaan ulang node.	Mulai ulang
XID74Kesalahan Nvidia	Ketentuan	Ada masalah dengan koneksi dari GPU ke GPU lain atau NVSwitch lebih. NVLink Ini mungkin mengindikasikan kegagalan perangkat keras dengan tautan itu sendiri atau mungkin mengindikasikan masalah dengan perangkat di ujung tautan yang jauh.	Ganti

Nama	Kepelikan	Deskripsi	Tindakan perbaikan
XID79Kesalahan Nvidia	Ketentuan	Driver GPU mencoba mengakses GPU melalui koneksi PCI Express dan menemukan bahwa GPU tidak dapat diakses.	Ganti
XID94Kesalahan Nvidia	Ketentuan	Ada kesalahan memori ECC.	Mulai ulang
XID95Kesalahan Nvidia	Ketentuan	Ada kesalahan memori ECC.	Mulai ulang
XID119Kesalahan Nvidia	Ketentuan	GSP habis waktu menanggapi permintaan RPC dari bit lain di driver.	Ganti
Kesalahan Nvidia XID12 0	Ketentuan	GSP telah merespons tepat waktu, tetapi dengan kesalahan.	Ganti
XID121Kesalahan Nvidia	Ketentuan	C2C adalah interkoneksi chip. Ini memungkinkan berbagi memori antara CPUs, akselerator, dan banyak lagi.	Ganti

Nama	Kepelikan	Deskripsi	Tindakan perbaikan
Kesalahan Nvidia XID14 0	Ketentuan	Driver GPU mungkin telah mengamati kesalahan yang tidak dapat diperbaiki dalam memori GPU, sedemikian rupa untuk mengganggu kemampuan driver GPU untuk menandai halaman untuk offlining halaman dinamis atau pemetaan ulang baris.	Ganti
NvidiaPageRetirement	Peristiwa	Pengemudi GPU telah menandai halaman memori untuk pensiun. Ini dapat terjadi jika ada kesalahan bit ganda tunggal atau dua kesalahan bit tunggal ditemui di alamat yang sama.	Tidak ada
Peringatan NvidiaXid [Kode]	Peristiwa	Setiap kejadian XIDs selain yang didefinisikan dalam daftar ini menghasilkan peristiwa ini.	Tidak ada

Nama	Kepelikan	Deskripsi	Tindakan perbaikan
DCGMError	Ketentuan	Koneksi ke proses host Data Center GPU Manager (DCGM) hilang atau tidak dapat dibuat.	Tidak ada
DCGMDiagnosticKesalahan	Ketentuan	Terjadi dikeluarkan menjalankan diagnostik aktif DCGM.	Tidak ada
DCGMDiagnosticKegagalan	Ketentuan	Kasus uji dari rangkaian uji diagnostik aktif DCGM gagal.	Tidak ada

Masalah kesehatan node runtime

Nama	Kepelikan	Deskripsi
PodStuckTerminating	Ketentuan	Sebuah Pod sedang atau macet terminating untuk waktu yang berlebihan, yang dapat disebabkan oleh kesalahan CRI yang mencegah perkembangan status pod.
%sRepeatedRestart	Peristiwa	Memulai ulang layanan systemd apa pun pada node (diformat menggunakan nama unit selubung judul).
ContainerRuntimeFailed	Peristiwa	Runtime container gagal membuat container, kemungkinan terkait dengan

Nama	Kepelikan	Deskripsi
		masalah yang dilaporkan jika terjadi berulang kali.
KubeletFailed	Peristiwa	Kubelet memasuki keadaan gagal.
LivenessProbeFailures	Peristiwa	Kegagalan probe keaktifan terdeteksi, berpotensi menunjukkan masalah kode aplikasi atau nilai batas waktu yang tidak mencukupi jika terjadi berulang kali.
ReadinessProbeFailures	Peristiwa	Kegagalan probe kesiapan terdeteksi, berpotensi menunjukkan masalah kode aplikasi atau nilai batas waktu yang tidak mencukupi jika terjadi berulang kali.
ServiceFailedToStart	Peristiwa	Unit systemd gagal memulai.

Masalah kesehatan simpul penyimpanan

Nama	Kepelikan	Deskripsi
XFSSmallAverageClusterSize	Ketentuan	Ukuran XFS Average Cluster kecil, menunjukkan fragmentasi ruang kosong yang berlebihan yang dapat mencegah pembuatan file meskipun ada inode atau ruang kosong yang tersedia.
EtcHostsMountFailed	Peristiwa	Pemasangan kubelet yang dihasilkan /etc/hosts

Nama	Kepelikan	Deskripsi
		gagal karena remounting /var/lib/kubelet/pods data pengguna selama operasi kubelet-container.
IODelays	Peristiwa	Penundaan input atau output terdeteksi dalam suatu proses, berpotensi menunjukkan penyediaan input-output yang tidak mencukupi jika berlebihan.
KubeletDiskUsageSlow	Peristiwa	Kubelet melaporkan penggunaan disk yang lambat saat mencoba mengakses sistem file, berpotensi menunjukkan masalah input-output atau sistem file disk yang tidak mencukupi.

Lihat status kesehatan node Anda

Topik ini menjelaskan alat dan metode yang tersedia untuk memantau status kesehatan simpul di kluster Amazon EKS. Informasi tersebut mencakup kondisi node, peristiwa, dan kasus deteksi yang membantu Anda mengidentifikasi dan mendiagnosis masalah tingkat simpul. Gunakan perintah dan pola yang dijelaskan di sini untuk memeriksa sumber daya kesehatan node, menafsirkan kondisi status, dan menganalisis peristiwa node untuk pemecahan masalah operasional.

Anda bisa mendapatkan beberapa informasi kesehatan node dengan perintah Kubernetes untuk semua node. Dan jika Anda menggunakan agen pemantauan node melalui Amazon EKS Auto Mode atau add-on terkelola Amazon EKS, Anda akan mendapatkan lebih banyak variasi sinyal node untuk membantu memecahkan masalah. Deskripsi masalah kesehatan yang terdeteksi oleh agen pemantauan simpul juga tersedia di dasbor observabilitas. Untuk informasi selengkapnya, lihat [the section called “Kesehatan simpul”](#).

Kondisi simpul

Kondisi node mewakili masalah terminal yang membutuhkan tindakan remediasi seperti penggantian instance atau reboot.

Untuk mendapatkan kondisi untuk semua node:

```
kubectl get nodes -o 'custom-  
columns=NAME:.metadata.name,CONDITIONS:.status.conditions[*].type,STATUS:.status.conditions[*].
```

Untuk mendapatkan kondisi rinci untuk node tertentu

```
kubectl describe node node-name
```

Contoh kondisi output dari node yang sehat:

```
- lastHeartbeatTime: "2024-11-21T19:07:40Z"  
  lastTransitionTime: "2024-11-08T03:57:40Z"  
  message: Monitoring for the Networking system is active  
  reason: NetworkingIsReady  
  status: "True"  
  type: NetworkingReady
```

Contoh kondisi node yang tidak sehat dengan masalah jaringan:

```
- lastHeartbeatTime: "2024-11-21T19:12:29Z"  
  lastTransitionTime: "2024-11-08T17:04:17Z"  
  message: IPAM-D has failed to connect to API Server which could be an issue with  
    IPTable rules or any other network configuration.  
  reason: IPAMNotReady  
  status: "False"  
  type: NetworkingReady
```

Peristiwa simpul

Peristiwa node menunjukkan masalah sementara atau konfigurasi sub-optimal.

Untuk mendapatkan semua peristiwa yang dilaporkan oleh agen pemantauan node

Ketika agen pemantauan node tersedia, Anda dapat menjalankan perintah berikut.


```
kubectl get events --field-selector=reportingComponent=eks-node-monitoring-agent
```

Contoh output:

LAST SEEN	TYPE	REASON	OBJECT
MESSAGE			
4s	Warning	SoftLockup	node/ip-192-168-71-251.us-west-2.compute.internal
CPU stuck for 23s			

Untuk mendapatkan acara untuk semua node

```
kubectl get events --field-selector involvedObject.kind=Node
```

Untuk mendapatkan acara untuk node tertentu

```
kubectl get events --field-selector involvedObject.kind=Node,involvedObject.name=node-name
```

Untuk menonton acara secara real-time

```
kubectl get events -w --field-selector involvedObject.kind=Node
```

Contoh keluaran acara:

LAST SEEN	TYPE	REASON	OBJECT	MESSAGE
2m	Warning	MemoryPressure	Node/node-1	Node experiencing memory pressure
5m	Normal	NodeReady	Node/node-1	Node became ready

Perintah pemecahan masalah umum

```
# Get comprehensive node status
kubectl get node node-name -o yaml

# Watch node status changes
kubectl get nodes -w

# Get node metrics
kubectl top node
```

Mengambil log node untuk node terkelola menggunakan kubectl dan S3

Pelajari cara mengambil log node untuk node terkelola Amazon EKS yang memiliki agen pemantauan node.

Prasyarat

Pastikan Anda memiliki yang berikut:

- Cluster Amazon EKS yang ada dengan agen pemantauan node. Untuk informasi selengkapnya, lihat [the section called “Kesehatan simpul”](#).
- Alat `kubectl` baris perintah diinstal dan dikonfigurasi untuk berkomunikasi dengan cluster Anda.
- AWS CLI diinstal dan masuk dengan izin yang cukup untuk membuat bucket dan objek S3.
- Versi terbaru dari Python 3 diinstal
- AWS SDK untuk Python 3, Boto 3, diinstal.

Langkah 1: Buat tujuan bucket S3 (opsional)

Jika Anda belum memiliki ember S3 untuk menyimpan log, buat satu. Gunakan perintah AWS CLI berikut. Bucket default ke daftar kontrol `private` akses. Ganti *bucket-name* dengan nama bucket unik pilihan Anda.

```
aws s3api create-bucket --bucket bucket-name
```

Langkah 2: Buat URL S3 yang telah ditandatangani sebelumnya untuk HTTP Put

Amazon EKS mengembalikan log node dengan melakukan operasi HTTP PUT ke URL yang Anda tentukan. Dalam tutorial ini, kita akan menghasilkan URL PUT HTTP S3 yang telah ditandatangani sebelumnya.

Log akan dikembalikan sebagai tarball gzip, dengan ekstensi. `.tar.gz`

Note

Anda harus menggunakan AWS API atau SDK untuk membuat URL unggahan S3 yang telah ditandatangani sebelumnya untuk EKS untuk mengunggah file log. Anda tidak dapat

membuat URL unggahan S3 yang telah ditandatangani sebelumnya menggunakan CLI AWS .

1. Tentukan di mana dalam ember Anda ingin menyimpan log. Misalnya, Anda dapat menggunakan `2024-11-12/logs1.tar.gz` sebagai kunci.
2. Simpan kode Python berikut ke file `presign-upload.py` Ganti `<bucket-name>` dan `<key>`. Kuncinya harus diakhiri dengan `.tar.gz`.

```
import boto3; print(boto3.client('s3').generate_presigned_url(
    ClientMethod='put_object',
    Params={'Bucket': '<bucket-name>', 'Key': '<key>'},
    ExpiresIn=1000
))
```

3. Jalankan skrip dengan

```
python presign-upload.py
```

4. Perhatikan output URL. Gunakan nilai ini di langkah berikutnya sebagai `http-put-destination`.

Untuk informasi selengkapnya, lihat [Menghasilkan URL yang telah ditetapkan sebelumnya untuk mengunggah file](#) di AWS Boto3 SDK untuk Dokumentasi Python.

Langkah 3: Buat NodeDiagnostic sumber daya

Identifikasi nama node tempat Anda ingin mengumpulkan log dari.

Buat NodeDiagnostic manifest yang menggunakan nama node sebagai nama sumber daya, dan berikan tujuan URL PUT HTTP.

```
apiVersion: eks.amazonaws.com/v1alpha1
kind: NodeDiagnostic
metadata:
  name: node-name
spec:
  logCapture:
    destination: http-put-destination
```

Menerapkan manifes ke kluster.

```
kubectl apply -f nodediagnostic.yaml
```

Anda dapat memeriksa Status koleksi dengan menjelaskan NodeDiagnostic sumber daya:

- Status Success atau SuccessWithErrors menunjukkan bahwa tugas selesai dan log yang diunggah ke tujuan yang disediakan (SuccessWithErrors menunjukkan bahwa beberapa log mungkin hilang)
- Jika statusnya Gagal, konfirmasi URL unggahan sudah terbentuk dengan baik dan tidak kedaluwarsa.

```
kubectl describe nodediagnostics.eks.amazonaws.com/node-name
```

Langkah 4: Unduh log dari S3

Tunggu sekitar satu menit sebelum mencoba mengunduh log. Kemudian, gunakan S3 CLI untuk mengunduh log.

```
# Once NodeDiagnostic shows Success status, download the logs  
aws s3 cp s3://bucket-name/key ./node-logs.tar.gz
```

Langkah 5: Bersihkan NodeDiagnostic sumber daya

- NodeDiagnostic sumber daya tidak dihapus secara otomatis. Anda harus membersihkannya sendiri setelah Anda mendapatkan artefak log Anda

```
# Delete the NodeDiagnostic resource  
kubectl delete nodediagnostics.eks.amazonaws.com/node-name
```

Ikhtisar Amazon EKS Hybrid Nodes

Dengan Amazon EKS Hybrid Nodes, Anda dapat menggunakan infrastruktur lokal dan edge sebagai node di kluster Amazon EKS. AWS mengelola bidang kontrol Kubernetes yang AWS di-host dari

klaster Amazon EKS, dan Anda mengelola node hibrida yang berjalan di lingkungan lokal atau edge Anda. Ini menyatukan manajemen Kubernetes di seluruh lingkungan Anda dan menurunkan manajemen bidang kontrol Kubernetes untuk aplikasi lokal dan edge Anda. AWS

Amazon EKS Hybrid Nodes bekerja dengan perangkat keras lokal atau mesin virtual apa pun, menghadirkan efisiensi, skalabilitas, dan ketersediaan Amazon EKS ke mana pun aplikasi Anda perlu dijalankan. Anda dapat menggunakan berbagai fitur Amazon EKS dengan Amazon EKS Hybrid Nodes termasuk add-on Amazon EKS, Amazon EKS Pod Identity, entri akses cluster, wawasan cluster, dan dukungan versi Kubernetes yang diperluas. Amazon EKS Hybrid Nodes terintegrasi secara native dengan AWS layanan termasuk AWS Systems Manager, AWS IAM Roles Anywhere, Amazon Managed Service for Prometheus, dan CloudWatch Amazon untuk pemantauan terpusat, pencatatan, dan manajemen identitas.

Dengan Amazon EKS Hybrid Nodes, tidak ada komitmen di muka atau biaya minimum, dan Anda dikenakan biaya per jam untuk sumber daya vCPU node hybrid Anda ketika mereka dilampirkan ke cluster Amazon EKS Anda. Untuk informasi harga selengkapnya, lihat [Harga Amazon EKS](#).

Fitur

EKS Hybrid Nodes memiliki fitur tingkat tinggi berikut:

- Bidang kontrol Kubernetes yang dikelola: AWS mengelola bidang kontrol Kubernetes yang AWS di-host dari klaster EKS, dan Anda mengelola node hibrida yang berjalan di lingkungan lokal atau edge Anda. Ini menyatukan manajemen Kubernetes di seluruh lingkungan Anda dan menurunkan manajemen bidang kontrol Kubernetes untuk aplikasi lokal dan edge Anda. AWS Dengan memindahkan control plane Kubernetes AWS, Anda dapat menghemat kapasitas lokal untuk aplikasi Anda dan percaya bahwa pesawat kontrol Kubernetes berskala dengan beban kerja Anda.
- Pengalaman EKS yang konsisten: Sebagian besar fitur EKS didukung dengan EKS Hybrid Nodes untuk pengalaman EKS yang konsisten di lingkungan lokal dan cloud Anda termasuk add-on EKS, EKS Pod Identity, entri akses klaster, wawasan cluster, dukungan versi Kubernetes yang diperluas, dan banyak lagi. Lihat [the section called “Konfigurasi add-on”](#) untuk informasi selengkapnya tentang add-on EKS yang didukung dengan EKS Hybrid Nodes.
- Observabilitas terpusat dan manajemen identitas: EKS Hybrid Nodes terintegrasi secara native dengan layanan termasuk AWS Systems Manager, AWS IAM Roles Anywhere, Amazon Managed Service for Prometheus, dan Amazon CloudWatch untuk pemantauan terpusat, logging, dan manajemen identitas.

- Burst-to-cloud atau tambahkan kapasitas lokal: Satu kluster EKS dapat digunakan untuk menjalankan node dan node hibrid di AWS Wilayah, AWS Local Zones, atau AWS Outposts burst-to-cloud ke atau menambahkan kapasitas lokal ke kluster EKS Anda. Lihat [Pertimbangan untuk cluster mode campuran untuk](#) informasi selengkapnya.
- Infrastruktur fleksibel: EKS Hybrid Nodes mengikuti pendekatan membawa infrastruktur Anda sendiri dan agnostik ke infrastruktur yang Anda gunakan untuk node hibrida. Anda dapat menjalankan node hybrid pada mesin fisik atau virtual, dan arsitektur x86 dan ARM, sehingga memungkinkan untuk memigrasikan beban kerja lokal yang berjalan pada node hybrid di berbagai jenis infrastruktur.
- Jaringan fleksibel: Dengan EKS Hybrid Nodes, komunikasi antara bidang kontrol EKS dan node hybrid dirutekan melalui VPC dan subnet yang Anda lewati selama pembuatan cluster, yang dibangun di atas mekanisme yang [ada di EKS untuk bidang kontrol ke](#) jaringan node. Ini fleksibel untuk metode pilihan Anda untuk menghubungkan jaringan lokal Anda ke AWS VPC di. Ada beberapa [opsi terdokumentasi](#) yang tersedia termasuk AWS Site-to-Site VPN, AWS Direct Connect, atau solusi VPN Anda sendiri, dan Anda dapat memilih metode yang paling sesuai dengan kasus penggunaan Anda.

Batas

- Hingga 15 CIDRs untuk Remote Node Networks dan 15 CIDRs untuk Remote Pod Networks per cluster didukung.

Pertimbangan

- EKS Hybrid Nodes dapat digunakan dengan cluster EKS baru atau yang sudah ada.
- EKS Hybrid Node tersedia di semua AWS Wilayah, kecuali Wilayah AWS GovCloud (AS) dan Wilayah AWS Tiongkok.
- EKS Hybrid Nodes harus memiliki koneksi yang andal antara lingkungan lokal Anda dan AWS. EKS Hybrid Nodes tidak cocok untuk lingkungan yang terputus, terganggu, intermiten atau terbatas (DDIL). Jika Anda menjalankan di lingkungan DDIL, pertimbangkan [Amazon EKS Anywhere](#). Referensi [Praktik Terbaik untuk Node Hibrid EKS](#) untuk informasi tentang bagaimana node hibrida berperilaku selama skenario pemutusan jaringan.
- Menjalankan EKS Hybrid Nodes pada infrastruktur cloud, termasuk AWS Wilayah, AWS Local Zones, AWS Outposts, atau di cloud lainnya, tidak didukung. Anda akan dikenakan biaya node hybrid jika Anda menjalankan node hybrid di EC2 instans Amazon.

- Penagihan untuk node hybrid dimulai ketika node bergabung dengan cluster EKS dan berhenti ketika node dihapus dari cluster. Pastikan untuk menghapus node hybrid Anda dari cluster EKS Anda jika Anda tidak menggunakannya.

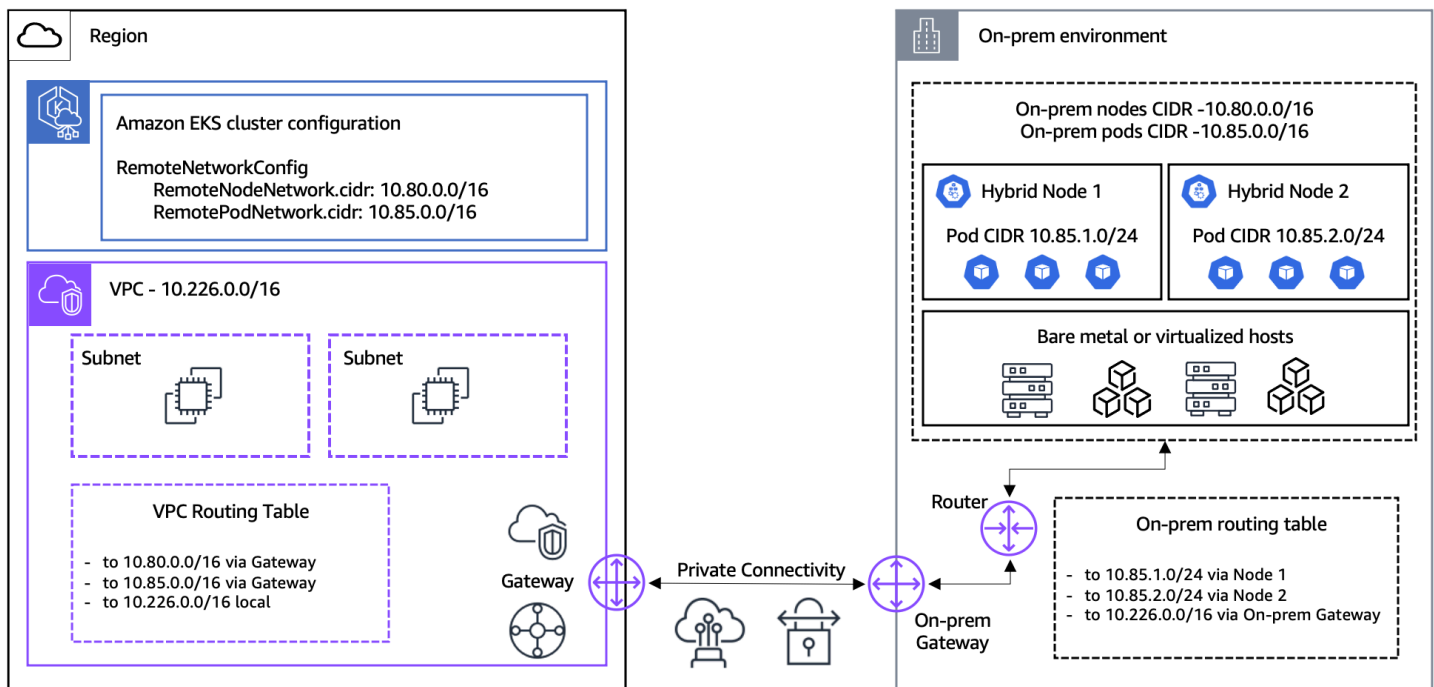
Sumber daya tambahan

- [Lokakarya EKS Hybrid Nodes](#): Step-by-step instruksi untuk menerapkan EKS Hybrid Nodes di lingkungan demo.
- [AWS re:invent: EKS Hybrid Nodes](#): AWS re:Invent session memperkenalkan peluncuran EKS Hybrid Nodes dengan pelanggan yang menunjukkan bagaimana mereka menggunakan EKS Hybrid Nodes di lingkungan mereka.
- [AWS re:post: Jaringan cluster untuk EKS Hybrid Nodes](#): Artikel yang menjelaskan berbagai metode untuk menyiapkan jaringan untuk EKS Hybrid Nodes.
- [AWS blog: Jalankan inferensi GenAI di seluruh lingkungan dengan EKS Hybrid Nodes](#): Posting blog yang menunjukkan cara menjalankan inferensi GenAI di seluruh lingkungan dengan EKS Hybrid Nodes.

Penyiapan prasyarat untuk node hybrid

Untuk menggunakan Amazon EKS Hybrid Nodes, Anda harus memiliki konektivitas pribadi dari lingkungan lokal ke/dari AWS, server bare metal atau mesin virtual dengan sistem operasi yang didukung, dan aktivasi hybrid AWS IAM Roles Anywhere atau AWS Systems Manager (SSM) yang dikonfigurasi. Anda bertanggung jawab untuk mengelola prasyarat ini di seluruh siklus hidup node hybrid.

- Konektivitas jaringan hybrid dari lingkungan lokal Anda ke/dari AWS
- Infrastruktur dalam bentuk mesin fisik atau virtual
- Sistem operasi yang kompatibel dengan node hybrid
- Penyedia kredensi IAM lokal dikonfigurasi



Konektivitas jaringan hybrid

Komunikasi antara bidang kontrol Amazon EKS dan node hybrid dirutekan melalui VPC dan subnet yang Anda lewati selama pembuatan cluster, yang dibangun di atas [mekanisme yang ada](#) di Amazon EKS untuk bidang kontrol ke jaringan node. Ada beberapa [opsi terdokumentasi](#) yang tersedia bagi Anda untuk menghubungkan lingkungan lokal Anda dengan VPC Anda AWS Site-to-Site termasuk VPN, AWS Direct Connect, atau koneksi VPN Anda sendiri. Referensikan panduan pengguna [AWS Site-to-Site VPN](#) dan [AWS Direct Connect](#) untuk informasi lebih lanjut tentang cara menggunakan solusi tersebut untuk koneksi jaringan hybrid Anda.

Untuk pengalaman yang optimal, kami menyarankan Anda memiliki konektivitas jaringan yang andal minimal 100 Mbps dan latensi pulang-pergi maksimum 200 ms untuk koneksi node hibrida ke Wilayah. AWS Ini adalah panduan umum yang mengakomodasi sebagian besar kasus penggunaan tetapi bukan persyaratan yang ketat. Persyaratan bandwidth dan latensi dapat bervariasi tergantung pada jumlah node hibrida dan karakteristik beban kerja Anda, seperti ukuran gambar aplikasi, elastisitas aplikasi, konfigurasi pemantauan dan logging, dan dependensi aplikasi untuk mengakses data yang disimpan di layanan lain. AWS Kami menyarankan Anda menguji dengan aplikasi dan lingkungan Anda sendiri sebelum menerapkan ke produksi untuk memvalidasi bahwa pengaturan jaringan Anda memenuhi persyaratan untuk beban kerja Anda.

Konfigurasi jaringan lokal

Anda harus mengaktifkan akses jaringan masuk dari bidang kontrol Amazon EKS ke lingkungan lokal agar bidang kontrol Amazon EKS dapat berkomunikasi dengan node hibrid yang kubelet berjalan dan secara opsional dengan webhook yang berjalan di node hybrid Anda. Selain itu, Anda harus mengaktifkan akses jaringan keluar untuk node hybrid dan komponen yang berjalan di dalamnya untuk berkomunikasi dengan bidang kontrol Amazon EKS. Anda dapat mengonfigurasi komunikasi ini agar tetap sepenuhnya pribadi dengan AWS Direct Connect, AWS Site-to-Site VPN, atau koneksi VPN Anda sendiri.

Rentang Classless Inter-Domain Routing (CIDR) yang Anda gunakan untuk node lokal dan jaringan pod harus menggunakan rentang alamat RFC-1918. IPv4 Router lokal Anda harus dikonfigurasi dengan rute ke node lokal dan pod opsional. Lihat [the section called “Konfigurasi jaringan lokal”](#) untuk informasi selengkapnya tentang persyaratan jaringan lokal, termasuk daftar lengkap port dan protokol yang diperlukan yang harus diaktifkan di firewall dan lingkungan lokal.

Konfigurasi kluster EKS

Untuk meminimalkan latensi, sebaiknya Anda membuat kluster Amazon EKS di AWS Wilayah yang paling dekat dengan lingkungan lokal atau edge Anda. Anda meneruskan node dan pod lokal CIDRs selama pembuatan kluster Amazon EKS melalui dua bidang API: `RemoteNodeNetwork` dan `RemotePodNetwork`. Anda mungkin perlu berdiskusi dengan tim jaringan lokal untuk mengidentifikasi node dan pod lokal. CIDRs Node CIDR dialokasikan dari jaringan lokal Anda dan pod CIDR dialokasikan dari Container Network Interface (CNI) yang Anda gunakan jika Anda menggunakan jaringan overlay untuk CNI Anda. Cilium dan Calico menggunakan jaringan overlay secara default.

Node dan pod lokal yang CIDRs Anda konfigurasikan melalui `RemotePodNetwork` bidang `RemoteNodeNetwork` dan digunakan untuk mengonfigurasi bidang kontrol Amazon EKS untuk merutekan lalu lintas melalui VPC Anda ke dan pod kubelet yang berjalan pada node hibrid Anda. Node dan pod lokal Anda CIDRs tidak dapat saling tumpang tindih, CIDR VPC yang Anda lewati selama pembuatan kluster, atau konfigurasi layanan IPv4 untuk kluster Amazon EKS Anda.

Kami menyarankan Anda menggunakan akses endpoint publik atau pribadi untuk endpoint server API Amazon EKS Kubernetes. Jika Anda memilih “Publik dan Pribadi”, endpoint server API Amazon EKS Kubernetes akan selalu menyelesaikan ke publik IPs untuk node hybrid yang berjalan di luar VPC Anda, yang dapat mencegah node hybrid Anda bergabung dengan cluster. Saat Anda menggunakan akses titik akhir publik, titik akhir server API Kubernetes diselesaikan ke publik IPs dan

komunikasi dari node hibrida ke bidang kontrol Amazon EKS akan dirutekan melalui internet. Ketika Anda memilih akses endpoint pribadi, titik akhir server Kubernetes API diselesaikan menjadi pribadi IPs dan komunikasi dari node hibrida ke bidang kontrol Amazon EKS akan dirutekan melalui tautan konektivitas pribadi Anda, dalam banyak kasus Direct Connect atau VPN. AWS Site-to-Site

Konfigurasi VPC

Anda harus mengonfigurasi VPC yang Anda lewati selama pembuatan kluster Amazon EKS dengan rute dalam tabel peruteannya untuk node lokal dan jaringan pod opsional dengan gateway pribadi virtual (VGW) atau gateway transit (TGW) sebagai target. Contoh ditunjukkan di bawah ini. Ganti `REMOTE_NODE_CIDR` dan `REMOTE_POD_CIDR` dengan nilai untuk jaringan lokal Anda.

Tujuan	Target	Deskripsi
10.226.0.0/16	lokal	Lalu lintas lokal ke rute VPC dalam VPC
<code>REMOTE_NODE_CIDR</code>	tgw-abcdef123456	Node CIDR on-prem, rute lalu lintas ke TGW
<code>REMOTE_POD_CIDR</code>	tgw-abcdef123456	Pod CIDR on-prem, rute lalu lintas ke TGW

Konfigurasi grup keamanan

Saat Anda membuat kluster, Amazon EKS membuat grup keamanan yang diberi nama `eks-cluster-sg-<cluster-name>-<uniqueID>`. Anda tidak dapat mengubah aturan masuk Grup Keamanan Cluster ini tetapi Anda dapat membatasi aturan keluar. Anda harus menambahkan grup keamanan tambahan ke kluster Anda untuk mengaktifkan kubelet dan webhook opsional yang berjalan pada node hybrid Anda untuk menghubungi control plane Amazon EKS. Aturan masuk yang diperlukan untuk grup keamanan tambahan ini ditunjukkan di bawah ini. Ganti `REMOTE_NODE_CIDR` dan `REMOTE_POD_CIDR` dengan nilai untuk jaringan lokal Anda.

Nama	ID aturan grup keamanan	Versi IP	Jenis	Protokol	Rentang port	Sumber
Node on-prem masuk	sgr-abcde f123456	IPv4	HTTPS	TCP	443	REMOTE_NODE_CIDR
Pod masuk di prem	sgr-abcde f654321	IPv4	HTTPS	TCP	443	REMOTE_POD_CIDR

Infrastruktur

Anda harus memiliki server bare metal atau mesin virtual yang tersedia untuk digunakan sebagai node hybrid. Node hybrid bersifat agnostik terhadap infrastruktur yang mendasarinya dan mendukung arsitektur x86 dan ARM. Amazon EKS Hybrid Nodes mengikuti pendekatan “bawa infrastruktur Anda sendiri”, di mana Anda bertanggung jawab untuk menyediakan dan mengelola server bare metal atau mesin virtual yang Anda gunakan untuk node hybrid. Meskipun tidak ada persyaratan sumber daya minimum yang ketat, kami menyarankan Anda menggunakan host dengan setidaknya 1 vCPU dan 1GiB RAM untuk node hybrid.

Sistem operasi

Bottlerocket, Amazon Linux 2023 AL2 (023), Ubuntu, dan RHEL divalidasi secara berkelanjutan untuk digunakan sebagai sistem operasi node untuk node hibrida. Bottlerocket didukung oleh di lingkungan AWS VMware vSphere saja. AL2023 tidak tercakup oleh AWS Support Plans saat dijalankan di luar Amazon EC2. AL2023 hanya dapat digunakan di lingkungan virtual lokal, lihat [Panduan Pengguna Amazon Linux 2023](#) untuk informasi selengkapnya. AWS mendukung integrasi node hybrid dengan sistem operasi Ubuntu dan RHEL tetapi tidak memberikan dukungan untuk sistem operasi itu sendiri.

Anda bertanggung jawab atas penyediaan dan manajemen sistem operasi. Saat Anda menguji node hybrid untuk pertama kalinya, paling mudah menjalankan Amazon EKS Hybrid Nodes CLI (nodeadm) pada host yang sudah disediakan. Untuk penerapan produksi, kami menyarankan Anda menyertakan nodeadm gambar sistem operasi emas Anda yang dikonfigurasi untuk dijalankan sebagai layanan systemd untuk secara otomatis bergabung dengan host ke kluster Amazon EKS saat startup host.

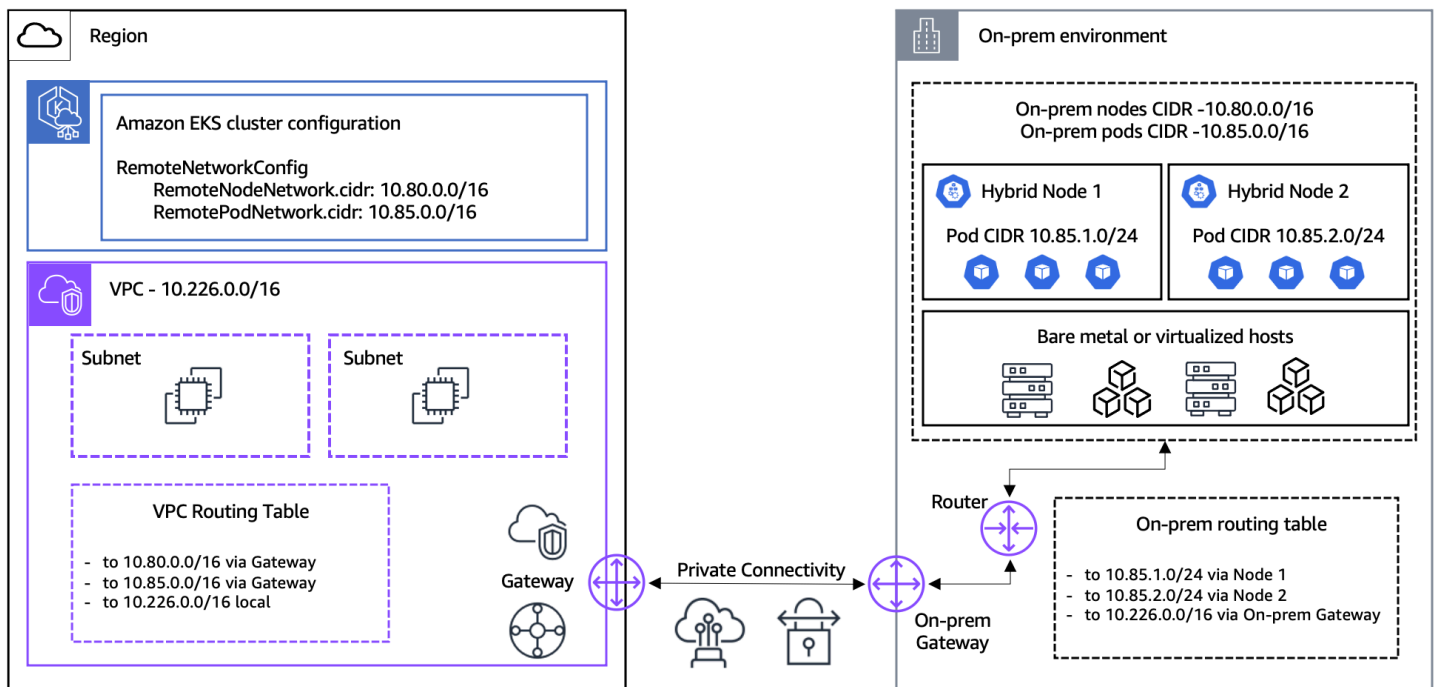
Penyedia kredensi IAM lokal

Amazon EKS Hybrid Nodes menggunakan kredensial IAM sementara yang disediakan oleh aktivasi hybrid AWS SSM atau AWS Peran IAM Anywhere untuk mengautentikasi dengan kluster Amazon EKS. Anda harus menggunakan aktivasi hibrida AWS SSM atau Peran AWS IAM Di Mana Saja dengan Amazon EKS Hybrid Nodes CLI (`nodeadm`). Kami menyarankan Anda menggunakan aktivasi hibrida AWS SSM jika Anda tidak memiliki Infrastruktur Kunci Publik (PKI) dengan Otoritas Sertifikat (CA) dan sertifikat untuk lingkungan lokal Anda. Jika Anda memiliki PKI dan sertifikat yang ada di lokasi, gunakan Peran AWS IAM Di Mana Saja.

Mirip dengan node [the section called “IAM role simpel”](#) for yang berjalan di Amazon EC2, Anda akan membuat Peran IAM Node Hybrid dengan izin yang diperlukan untuk menggabungkan node hybrid ke kluster Amazon EKS. Jika Anda menggunakan Peran AWS IAM Di Mana Saja, konfigurasi kebijakan kepercayaan yang memungkinkan AWS IAM Roles Anywhere untuk mengambil Peran IAM Node Hybrid dan mengonfigurasi profil IAM Roles Anywhere Anda dengan Peran AWS IAM Node Hybrid sebagai peran yang dapat diasumsikan. Jika Anda menggunakan AWS SSM, konfigurasi kebijakan kepercayaan yang memungkinkan AWS SSM untuk mengasumsikan Peran IAM Node Hybrid dan buat aktivasi hibrida dengan Peran IAM Node Hybrid. Lihat [the section called “Siapkan kredensial”](#) cara membuat Peran IAM Node Hybrid dengan izin yang diperlukan.

Mempersiapkan jaringan untuk node hybrid

Topik ini memberikan gambaran umum tentang pengaturan jaringan yang harus Anda konfigurasi sebelum membuat kluster Amazon EKS dan melampirkan node hybrid. Panduan ini mengasumsikan Anda telah memenuhi persyaratan prasyarat untuk konektivitas jaringan hybrid menggunakan VPN [AWS Site-to-Site](#) , [Direct AWS Connect](#), atau solusi VPN Anda sendiri.



Konfigurasi jaringan lokal

Persyaratan jaringan minimum

Untuk pengalaman yang optimal, kami menyarankan Anda memiliki konektivitas jaringan yang andal minimal 100 Mbps dan latensi pulang-pergi maksimum 200 ms untuk koneksi node hibrida ke Wilayah. AWS Ini adalah panduan umum yang mengakomodasi sebagian besar kasus penggunaan tetapi bukan persyaratan yang ketat. Persyaratan bandwidth dan latensi dapat bervariasi tergantung pada jumlah node hibrida dan karakteristik beban kerja Anda, seperti ukuran gambar aplikasi, elastisitas aplikasi, konfigurasi pemantauan dan logging, dan dependensi aplikasi untuk mengakses data yang disimpan di layanan lain. AWS Kami menyarankan Anda menguji dengan aplikasi dan lingkungan Anda sendiri sebelum menerapkan ke produksi untuk memvalidasi bahwa pengaturan jaringan Anda memenuhi persyaratan untuk beban kerja Anda.

Node dan pod lokal CIDRs

Identifikasi node dan pod yang akan CIDRs Anda gunakan untuk node hybrid Anda dan beban kerja yang berjalan di dalamnya. Node CIDR dialokasikan dari jaringan lokal Anda dan pod CIDR dialokasikan dari Container Network Interface (CNI) jika Anda menggunakan jaringan overlay untuk CNI Anda. Anda meneruskan node CIDRs dan pod lokal CIDRs sebagai input saat membuat kluster EKS dengan bidang `RemoteNodeNetwork` dan `RemotePodNetwork`. Node lokal Anda CIDRs harus dapat dirutekan di jaringan lokal Anda. Lihat bagian berikut untuk informasi tentang routabilitas CIDR pod lokal.

Blok CIDR node dan pod lokal harus memenuhi persyaratan berikut:

1. Berada dalam salah satu rentang IPv4 RFC-1918 berikut: 10.0.0.0/8,, atau. 172.16.0.0/12
192.168.0.0/16
2. Tidak saling tumpang tindih, CIDR VPC untuk kluster EKS Anda, atau CIDR layanan Kubernetes Anda. IPv4

Perutean jaringan pod lokal

Saat menggunakan EKS Hybrid Nodes, kami biasanya menyarankan agar Pod lokal CIDRs dapat dirutekan di jaringan lokal untuk mengaktifkan komunikasi dan fungsionalitas kluster lengkap antara lingkungan cloud dan lokal.

Jaringan pod yang dapat dirutekan

Jika Anda dapat membuat jaringan pod dapat dirutekan di jaringan lokal, ikuti panduan di bawah ini.

1. Konfigurasi RemotePodNetwork bidang untuk kluster EKS Anda dengan CIDR pod lokal, tabel rute VPC Anda dengan CIDR pod lokal, dan grup keamanan kluster EKS Anda dengan CIDR pod lokal Anda.
2. Ada beberapa teknik yang dapat Anda gunakan untuk membuat pod lokal CIDR dapat dirutekan di jaringan lokal Anda termasuk Border Gateway Protocol (BGP), rute statis, atau solusi perutean kustom lainnya. BGP adalah solusi yang direkomendasikan karena lebih skalabel dan lebih mudah dikelola daripada solusi alternatif yang memerlukan konfigurasi rute khusus atau manual. AWS mendukung kemampuan BGP Cilium dan Calico untuk pod iklan CIDRs, lihat [the section called “Konfigurasi CNI”](#) dan untuk informasi lebih lanjut. [the section called “Pod jarak jauh yang dapat dirutekan CIDRs”](#)
3. Webhook dapat berjalan pada node hybrid karena bidang kontrol EKS dapat berkomunikasi dengan alamat IP Pod yang ditetapkan ke webhooks.
4. Beban kerja yang berjalan pada node cloud dapat berkomunikasi secara langsung dengan beban kerja yang berjalan pada node hybrid di cluster EKS yang sama.
5. AWS Layanan lain, seperti AWS Application Load Balancers dan Amazon Managed Service for Prometheus, dapat berkomunikasi dengan beban kerja yang berjalan pada node hybrid untuk menyeimbangkan lalu lintas jaringan dan mengikis metrik pod.

Jaringan pod yang tidak dapat dirutekan

Jika Anda tidak dapat membuat jaringan pod Anda dapat dirutekan di jaringan lokal, ikuti panduan di bawah ini.

1. Webhook tidak dapat berjalan pada node hybrid karena webhook memerlukan konektivitas dari bidang kontrol EKS ke alamat IP Pod yang ditetapkan ke webhook. Dalam hal ini, kami menyarankan Anda menjalankan webhook pada node cloud di cluster EKS yang sama dengan node hybrid Anda, lihat [the section called “Konfigurasi webhook”](#) untuk informasi selengkapnya.
2. Beban kerja yang berjalan pada node cloud tidak dapat berkomunikasi secara langsung dengan beban kerja yang berjalan pada node hybrid saat menggunakan VPC CNI untuk node cloud dan Cilium atau Calico untuk node hybrid.
3. Gunakan Distribusi Lalu Lintas Layanan untuk menjaga lalu lintas lokal ke zona asalnya. Untuk informasi selengkapnya tentang Distribusi Lalu Lintas Layanan, lihat [the section called “Konfigurasi Distribusi Lalu Lintas Layanan”](#).
4. Konfigurasi CNI Anda untuk menggunakan penyamaran jalan keluar atau terjemahan alamat jaringan (NAT) untuk lalu lintas pod karena meninggalkan host lokal Anda. Ini diaktifkan secara default di Cilium. Calico `natOutgoing` harus diatur ke `true`.
5. AWS Layanan lain, seperti AWS Application Load Balancers dan Amazon Managed Service untuk Prometheus, tidak dapat berkomunikasi dengan beban kerja yang berjalan pada node hybrid.

Akses diperlukan selama instalasi dan peningkatan node hybrid

Anda harus memiliki akses ke domain berikut selama proses instalasi di mana Anda menginstal dependensi node hybrid pada host Anda. Proses ini dapat dilakukan sekali ketika Anda sedang membangun gambar sistem operasi Anda atau dapat dilakukan pada setiap host saat runtime. Ini termasuk instalasi awal dan ketika Anda meng-upgrade versi Kubernetes dari node hybrid Anda.

Komponen	URL	Protokol	Port
Artefak simpul EKS (S3)	<code>https://hybrid-assets.eks.amazonaws.com</code>	HTTPS	443
Titik akhir layanan EKS	<code>https://eks. <i>region</i>.amazonaws.com</code>	HTTPS	443

Komponen	URL	Protokol	Port
Titik akhir layanan ECR	https://api.ecr. <i>region</i> .amazonaws.com	HTTPS	443
Titik akhir EKS ECR	Lihat the section called “Lihat pendaftar gambar Amazon” untuk titik akhir regional.	HTTPS	443
Titik akhir biner SSM 1	https://amazon-ssm - <i>region</i> .s3. <i>region</i> .amazonaws.com	HTTPS	443
Titik akhir layanan SSM 1	https://ssm. <i>region</i> .amazonaws.com	HTTPS	443
Titik akhir biner IAM Anywhere 2	https://rolesanywhere.amazonaws.com	HTTPS	443
Titik akhir layanan IAM Anywhere 2	https://rolesanywhere. <i>region</i> .amazonaws.com	HTTPS	443

 Note

¹ Akses ke titik akhir AWS SSM hanya diperlukan jika Anda menggunakan aktivasi hibrida AWS SSM untuk penyedia kredensi IAM lokal Anda.

² Akses ke titik akhir AWS IAM hanya diperlukan jika Anda menggunakan Peran IAM Di Mana Saja untuk penyedia kredensi AWS IAM lokal Anda.

Akses diperlukan untuk operasi klaster yang sedang berlangsung

Akses jaringan berikut untuk firewall lokal Anda diperlukan untuk operasi klaster yang sedang berlangsung.

 Important

Tergantung pada pilihan CNI Anda, Anda perlu mengkonfigurasi aturan akses jaringan tambahan untuk port CNI. Lihat dokumentasi [Cilium dan dokumentasi Calico](#) untuk detailnya.

Jenis	Protokol	Arahan	Port	Sumber	Tujuan	Penggunaan
HTTPS	TCP	Ke luar	443	CIDR Node Jarak Jauh	Kluster EKS IPs ¹	kubelet ke server API Kubernetes
HTTPS	TCP	Ke luar	443	CIDR Pod Jarak Jauh	Kluster EKS IPs ¹	Pod ke server API Kubernetes
HTTPS	TCP	Ke luar	443	CIDR Node Jarak Jauh	Titik akhir layanan SSM	Aktivasi hibrida SSM menyegarkan kredensial dan detak jantung SSM setiap 5 menit
HTTPS	TCP	Ke luar	443	CIDR Node Jarak Jauh	Titik akhir layanan IAM Anywhere	Peran IAM Di Mana Saja penyegaran kredensial

Jenis	Protokol	Arahan	Port	Sumber	Tujuan	Penggunaan
HTTPS	TCP	Ke luar	443	CIDR Pod Jarak Jauh	Titik Akhir Regional STS	Pod ke titik akhir STS, hanya diperlukan untuk IRSA
HTTPS	TCP	Ke luar	443	CIDR Node Jarak Jauh	Titik akhir layanan Amazon EKS Auth	Node ke titik akhir Amazon EKS Auth, hanya diperlukan untuk Amazon EKS Pod Identity
HTTPS	TCP	Ke dalam	10250	Kluster EKS IPs ¹	CIDR Node Jarak Jauh	Server API Kubernetes ke kubelet
HTTPS	TCP	Ke dalam	Port webhook	Kluster EKS IPs ¹	CIDR Pod Jarak Jauh	Server API Kubernetes ke webhook

Jenis	Protokol	Arahan	Port	Sumber	Tujuan	Penggunaa n
HTTPS	TCP, UDP	Masuk, Keluar	53	CIDR Pod Jarak Jauh	CIDR Pod Jarak Jauh	Pod ke CoreDNS. Jika Anda menjalank an setidakny a 1 replika CoreDNS di cloud, Anda harus mengizink an lalu lintas DNS ke VPC tempat CoreDNS berjalan.
Ditentukan pengguna	Ditentukan pengguna	Masuk, Keluar	Port aplikasi	CIDR Pod Jarak Jauh	CIDR Pod Jarak Jauh	Pod ke Pod

 Note

¹ IPs Kluster EKS. Lihat bagian berikut tentang antarmuka jaringan elastis Amazon EKS.

Antarmuka jaringan Amazon EKS

Amazon EKS melampirkan antarmuka jaringan ke subnet di VPC yang Anda lewati selama pembuatan cluster untuk mengaktifkan komunikasi antara bidang kontrol EKS dan VPC Anda. Antarmuka jaringan yang dibuat Amazon EKS dapat ditemukan setelah pembuatan cluster di EC2 konsol Amazon atau dengan AWS CLI. Antarmuka jaringan asli dihapus dan antarmuka jaringan baru dibuat ketika perubahan diterapkan pada kluster EKS Anda, seperti upgrade versi Kubernetes. Anda dapat membatasi rentang IP untuk antarmuka jaringan Amazon EKS dengan menggunakan ukuran

subnet terbatas untuk subnet yang Anda lewati selama pembuatan klaster, yang memudahkan konfigurasi firewall lokal Anda untuk memungkinkan inbound/outbound konektivitas ke kumpulan yang diketahui dan dibatasi ini. IPs Untuk mengontrol antarmuka jaringan subnet mana yang dibuat, Anda dapat membatasi jumlah subnet yang Anda tentukan saat membuat cluster atau Anda dapat memperbarui subnet setelah membuat cluster.

Antarmuka jaringan yang disediakan oleh Amazon EKS memiliki deskripsi formatnya. Amazon EKS *your-cluster-name* Lihat contoh di bawah ini untuk perintah AWS CLI yang dapat Anda gunakan untuk menemukan alamat IP antarmuka jaringan yang disediakan Amazon EKS. Ganti VPC_ID dengan ID VPC yang Anda lewati selama pembuatan cluster.

```
aws ec2 describe-network-interfaces \
--query 'NetworkInterfaces[?(VpcId == VPC_ID && contains(Description,Amazon
EKS))].PrivateIpAddress'
```

AWS Pengaturan VPC dan subnet

Persyaratan [VPC dan subnet yang ada untuk](#) Amazon EKS berlaku untuk cluster dengan node hybrid. Selain itu, CIDR VPC Anda tidak dapat tumpang tindih dengan node dan pod lokal Anda. CIDRs Anda harus mengonfigurasi rute di tabel perutean VPC untuk node lokal dan pod opsional. CIDRs Rute ini harus diatur untuk mengarahkan lalu lintas ke gateway yang Anda gunakan untuk konektivitas jaringan hybrid Anda, yang biasanya merupakan gateway pribadi virtual (VGW) atau gateway transit (TGW). Jika Anda menggunakan TGW atau VGW untuk menghubungkan VPC dengan lingkungan lokal, Anda harus membuat lampiran TGW atau VGW untuk VPC Anda. VPC Anda harus memiliki nama host DNS dan dukungan resolusi DNS.

Langkah-langkah berikut menggunakan AWS CLI. Anda juga dapat membuat sumber daya ini di AWS Management Console atau dengan antarmuka lain seperti AWS CloudFormation, AWS CDK, atau Terraform.

Langkah 1: Buat VPC

1. Jalankan perintah berikut untuk membuat VPC. Ganti VPC_CIDR dengan rentang IPv4 CIDR RFC-1918 (pribadi) atau non-RFC-1918 (publik) (misalnya). 10.0.0.0/16 Catatan: Resolusi DNS, yang merupakan persyaratan EKS, diaktifkan untuk VPC secara default.

```
aws ec2 create-vpc --cidr-block VPC_CIDR
```

2. Aktifkan nama host DNS untuk VPC Anda. Catatan, resolusi DNS diaktifkan untuk VPC secara default. Ganti VPC_ID dengan ID VPC yang Anda buat pada langkah sebelumnya.

```
aws ec2 modify-vpc-attribute --vpc-id VPC_ID --enable-dns-hostnames
```

Langkah 2: Buat subnet

Buat setidaknya 2 subnet. Amazon EKS menggunakan subnet ini untuk antarmuka jaringan cluster. Untuk informasi selengkapnya, lihat [Persyaratan dan pertimbangan Subnet](#).

1. Anda dapat menemukan zona ketersediaan untuk AWS Wilayah dengan perintah berikut. Ganti us-west-2 dengan wilayah Anda.

```
aws ec2 describe-availability-zones \  
  --query 'AvailabilityZones[?(RegionName == us-west-2)].ZoneName'
```

2. Buat subnet. Ganti VPC_ID dengan ID VPC. Ganti SUBNET_CIDR dengan blok CIDR untuk subnet Anda (misalnya 10.0.1.0/24). Ganti AZ dengan zona ketersediaan tempat subnet akan dibuat (misalnya us-west-2a). Subnet yang Anda buat harus berada di setidaknya 2 zona ketersediaan yang berbeda.

```
aws ec2 create-subnet \  
  --vpc-id VPC_ID \  
  --cidr-block SUBNET_CIDR \  
  --availability-zone AZ
```

(Opsional) Langkah 3: Lampirkan VPC dengan Amazon VPC Transit Gateway (TGW) AWS atau gateway pribadi virtual Direct Connect (VGW)

Jika Anda menggunakan TGW atau VGW, lampirkan VPC Anda ke TGW atau VGW. [Untuk informasi selengkapnya, lihat lampiran VPC Amazon di Gateway Transit VPC Amazon atau asosiasi gateway pribadi virtual Direct AWS Connect.](#)

Transit Gateway

Jalankan perintah berikut untuk melampirkan Transit Gateway. Ganti VPC_ID dengan ID VPC. Ganti SUBNET_ID1 dan SUBNET_ID2 dengan subnet yang Anda buat pada langkah sebelumnya. IDs Ganti TGW_ID dengan ID TGW Anda.

```
aws ec2 create-transit-gateway-vpc-attachment \  
  --vpc-id VPC_ID \  
  --subnet-ids SUBNET_ID1 SUBNET_ID2 \  
  --transit-gateway-id TGW_ID
```

Gerbang Pribadi Virtual

Jalankan perintah berikut untuk melampirkan Transit Gateway. Ganti *VPN_ID* dengan ID VGW Anda. Ganti *VPC_ID* dengan ID VPC.

```
aws ec2 attach-vpn-gateway \  
  --vpn-gateway-id VPN_ID \  
  --vpc-id VPC_ID
```

(Opsional) Langkah 4: Buat tabel rute

Anda dapat memodifikasi tabel rute utama untuk VPC atau Anda dapat membuat tabel rute khusus. Langkah-langkah berikut membuat tabel rute kustom dengan rute ke node dan pod CIDRs lokal. Untuk informasi selengkapnya, lihat [Tabel rute subnet](#). Ganti *VPC_ID* dengan ID VPC.

```
aws ec2 create-route-table --vpc-id VPC_ID
```

Langkah 5: Buat rute untuk node dan pod lokal

Buat rute dalam tabel rute untuk setiap node jarak jauh lokal Anda. Anda dapat memodifikasi tabel rute utama untuk VPC atau menggunakan tabel rute khusus yang Anda buat di langkah sebelumnya.

Contoh di bawah ini menunjukkan cara membuat rute untuk node dan pod CIDRs lokal Anda. Dalam contoh, gateway transit (TGW) digunakan untuk menghubungkan VPC dengan lingkungan lokal. Jika Anda memiliki beberapa node dan pod lokal CIDRs, ulangi langkah-langkah untuk setiap CIDR.

- Jika Anda menggunakan gateway internet atau virtual private gateway (VGW) ganti `--transit-gateway-id` dengan `--gateway-id`
- Ganti *RT_ID* dengan ID tabel rute yang Anda buat pada langkah sebelumnya.
- Ganti *REMOTE_NODE_CIDR* dengan rentang CIDR yang akan Anda gunakan untuk node hybrid Anda.

- Ganti `REMOTE_POD_CIDR` dengan rentang CIDR yang akan Anda gunakan untuk pod yang berjalan pada node hybrid. Rentang pod CIDR sesuai dengan konfigurasi Container Networking Interface (CNI), yang paling sering menggunakan jaringan overlay lokal. Untuk informasi selengkapnya, lihat [the section called “Konfigurasi CNI”](#).
- Ganti `TGW_ID` dengan ID TGW Anda.

Jaringan simpul jarak jauh

```
aws ec2 create-route \
  --route-table-id RT_ID \
  --destination-cidr-block REMOTE_NODE_CIDR \
  --transit-gateway-id TGW_ID
```

Jaringan Pod jarak jauh

```
aws ec2 create-route \
  --route-table-id RT_ID \
  --destination-cidr-block REMOTE_POD_CIDR \
  --transit-gateway-id TGW_ID
```

(Opsional) Langkah 6: Kaitkan subnet dengan tabel rute

Jika Anda membuat tabel rute kustom pada langkah sebelumnya, kaitkan setiap subnet yang Anda buat pada langkah sebelumnya dengan tabel rute kustom Anda. Jika Anda memodifikasi tabel rute utama VPC, subnet secara otomatis dikaitkan dengan tabel rute utama VPC dan Anda dapat melewati langkah ini.

Jalankan perintah berikut untuk setiap subnet yang Anda buat di langkah sebelumnya. Ganti `RT_ID` dengan tabel rute yang Anda buat pada langkah sebelumnya. Ganti `SUBNET_ID` dengan ID subnet.

```
aws ec2 associate-route-table --route-table-id RT_ID --subnet-id SUBNET_ID
```

Konfigurasi grup keamanan cluster

Akses berikut untuk grup keamanan klaster EKS Anda diperlukan untuk operasi klaster yang sedang berlangsung.

Jenis	Protokol	Arahan	Port	Sumber	Tujuan	Penggunaa n
HTTPS	TCP	Ke dalam	443	CIDR Node Jarak Jauh	N/A	Kubelet ke Kubernetes API server
HTTPS	TCP	Ke dalam	443	CIDR Pod Jarak Jauh	N/A	Pod yang membutuhk an akses ke server API K8s saat CNI tidak menggunak an NAT untuk lalu lintas pod.
HTTPS	TCP	Ke luar	10250	N/A	CIDR Node Jarak Jauh	Server API Kubernetes ke Kubelet
HTTPS	TCP	Ke luar	Port webhook	N/A	CIDR Pod Jarak Jauh	Kubernete s API server ke webhook (jika menjalank an webhook pada node hybrid)

Untuk membuat grup keamanan dengan aturan akses masuk, jalankan perintah berikut. Grup keamanan ini harus diteruskan saat Anda membuat cluster Amazon EKS Anda. Secara default,

perintah di bawah ini membuat grup keamanan yang memungkinkan semua akses keluar. Anda dapat membatasi akses keluar untuk menyertakan hanya aturan di atas. Jika Anda mempertimbangkan untuk membatasi aturan keluar, sebaiknya Anda menguji secara menyeluruh semua aplikasi dan konektivitas pod sebelum menerapkan aturan yang diubah ke cluster produksi.

- Pada perintah pertama, ganti `SG_NAME` dengan nama untuk grup keamanan Anda
- Pada perintah pertama, ganti `VPC_ID` dengan ID VPC yang Anda buat pada langkah sebelumnya
- Pada perintah kedua, ganti `SG_ID` dengan ID grup keamanan yang Anda buat di perintah pertama
- Pada perintah kedua, ganti `REMOTE_NODE_CIDR` dan `REMOTE_POD_CIDR` dengan nilai untuk node hybrid dan jaringan lokal Anda.

```
aws ec2 create-security-group \  
  --group-name SG_NAME \  
  --description "security group for hybrid nodes" \  
  --vpc-id VPC_ID
```

```
aws ec2 authorize-security-group-ingress \  
  --group-id SG_ID \  
  --ip-permissions '[{"IpProtocol": "tcp", "FromPort": 443, "ToPort": 443,  
  "IpRanges": [{"CidrIp": "REMOTE_NODE_CIDR"}, {"CidrIp": "REMOTE_POD_CIDR"}]]'
```

Siapkan sistem operasi untuk node hybrid

Bottlerocket, Amazon Linux 2023 AL2 (023), Ubuntu, dan RHEL divalidasi secara berkelanjutan untuk digunakan sebagai sistem operasi node untuk node hibrida. Bottlerocket didukung oleh di lingkungan AWS VMware vSphere saja. AL2023 tidak tercakup oleh AWS Support Plans saat dijalankan di luar Amazon EC2. AL2023 hanya dapat digunakan di lingkungan virtual lokal, lihat [Panduan Pengguna Amazon Linux 2023](#) untuk informasi selengkapnya. AWS mendukung integrasi node hybrid dengan sistem operasi Ubuntu dan RHEL tetapi tidak memberikan dukungan untuk sistem operasi itu sendiri.

Anda bertanggung jawab atas penyediaan dan manajemen sistem operasi. Saat Anda menguji node hybrid untuk pertama kalinya, paling mudah menjalankan Amazon EKS Hybrid Nodes CLI (nodeadm) pada host yang sudah disediakan. Untuk penerapan produksi, kami menyarankan Anda menyertakan nodeadm gambar sistem operasi yang dikonfigurasi untuk dijalankan sebagai layanan systemd untuk secara otomatis bergabung dengan host ke kluster Amazon EKS saat startup host. Jika Anda menggunakan Bottlerocket sebagai sistem operasi node Anda di vSphere, Anda tidak perlu

menggunakan nodeadm karena Bottlerocket sudah berisi dependensi yang diperlukan untuk node hybrid dan secara otomatis akan terhubung ke cluster yang Anda konfigurasi saat startup host.

Kompatibilitas versi

Tabel di bawah ini mewakili versi sistem operasi yang kompatibel dan divalidasi untuk digunakan sebagai sistem operasi node untuk node hybrid. Jika Anda menggunakan varian atau versi sistem operasi lain yang tidak termasuk dalam tabel ini, maka kompatibilitas node hybrid dengan varian atau versi sistem operasi Anda tidak tercakup oleh AWS Support. Node hybrid bersifat agnostik terhadap infrastruktur yang mendasarinya dan mendukung arsitektur x86 dan ARM.

Sistem Operasi	Versi
Amazon Linux	Amazon Linux 2023 (AL2023)
Bottlerocket	v1.37.0 dan VMware varian di atasnya yang menjalankan Kubernetes v1.28 dan di atasnya
Ubuntu	Ubuntu 20.04, Ubuntu 22.04, Ubuntu 24.04
Linux Red Hat Enterprise	RHEL 8, RHEL 9

Pertimbangan sistem operasi

Umum

- Amazon EKS Hybrid Nodes CLI (nodeadm) dapat digunakan untuk menyederhanakan instalasi dan konfigurasi komponen dan dependensi node hybrid. Anda dapat menjalankan nodeadm `install` proses selama pipeline build image sistem operasi atau saat runtime di setiap host lokal. Untuk informasi selengkapnya tentang komponen yang nodeadm diinstal, lihat [the section called “Node hibrida nodeadm”](#)
- Jika Anda menggunakan proxy di lingkungan lokal untuk menjangkau internet, ada konfigurasi sistem operasi tambahan yang diperlukan untuk proses penginstalan dan pemutakhiran guna mengonfigurasi manajer paket Anda agar menggunakan proxy. Lihat [the section called “Konfigurasi proxy”](#) untuk instruksi.

Bottlerocket

- Langkah-langkah dan alat untuk menghubungkan node Bottlerocket berbeda dari langkah-langkah untuk sistem operasi lain dan dibahas secara terpisah [the section called “Hubungkan node hybrid dengan Bottlerocket”](#), bukan langkah-langkah masuk. [the section called “Connect node hybrid”](#)
- Langkah-langkah untuk Bottlerocket tidak menggunakan alat CLI node hybrid, `nodeadm`
- Hanya VMware varian Bottlerocket versi v1.37.0 ke atas yang didukung dengan EKS Hybrid Nodes. VMware varian Bottlerocket tersedia untuk Kubernetes versi v1.28 ke atas. [Varian Bottlerocket lainnya](#) tidak didukung sebagai sistem operasi node hybrid. CATATAN: VMware varian Bottlerocket hanya tersedia untuk arsitektur x86_64.

Kontainer

- Containerd adalah runtime container Kubernetes standar dan merupakan dependensi untuk node hybrid, serta semua tipe komputasi node Amazon EKS. Amazon EKS Hybrid Nodes CLI (`nodeadm`) mencoba menginstal containerd selama proses berlangsung. `nodeadm install` Anda dapat mengonfigurasi instalasi containerd `nodeadm install` saat runtime dengan `--containerd-source` opsi baris perintah. Opsi yang valid adalah `none`, `distro`, dan `docker`. Jika Anda menggunakan RHEL, `distro` ini bukan opsi yang valid dan Anda dapat mengonfigurasi `nodeadm` untuk menginstal build containerd dari repo Docker atau Anda dapat menginstal containerd secara manual. Saat menggunakan AL2 023 atau Ubuntu, `nodeadm` default untuk menginstal containerd dari distribusi sistem operasi. Jika Anda tidak ingin `nodeadm` menginstal containerd, gunakan opsi. `--containerd-source none`

Ubuntu

- Jika Anda menggunakan Ubuntu 20.04, Anda harus menggunakan aktivasi hybrid AWS Systems Manager sebagai penyedia kredensi Anda. AWS IAM Roles Anywhere tidak didukung di Ubuntu 20.04.
- [Jika Anda menggunakan Ubuntu 24.04, Anda mungkin perlu memperbarui versi containerd atau mengubah konfigurasi AppArmor Anda untuk mengadopsi perbaikan yang memungkinkan pod dihentikan dengan benar, lihat Ubuntu #2065423.](#) Diperlukan reboot untuk menerapkan perubahan pada AppArmor profil. Versi terbaru Ubuntu 24.04 memiliki versi containerd yang diperbarui di manajer paketnya dengan perbaikan (containerd versi 1.7.19+).

RHEL

- Jika Anda menggunakan RHEL 8, Anda harus menggunakan aktivasi hybrid AWS Systems Manager sebagai penyedia kredensi Anda. AWS IAM Roles Anywhere tidak didukung di RHEL 8.

LENGAN

- Jika Anda menggunakan perangkat keras ARM, prosesor yang sesuai dengan ARMv8 .2 dengan Ekstensi Kriptografi (ARMv8.2+crypto) diperlukan untuk menjalankan add-on EKS kube-proxy versi 1.31 ke atas. Semua sistem Raspberry Pi sebelum Raspberry Pi 5, serta prosesor berbasis Cortex-A72, tidak memenuhi persyaratan ini. Sebagai solusinya, Anda dapat terus menggunakan add-on EKS kube-proxy versi 1.30 hingga mencapai akhir dukungan yang diperluas pada Juli 2026, lihat [the section called “Kalender rilis Amazon EKS Kubernetes”](#), atau gunakan gambar kube-proxy khusus dari hulu.
- Pesan kesalahan berikut di log kube-proxy menunjukkan ketidakcocokan ini:

```
Fatal glibc error: This version of Amazon Linux requires a newer ARM64 processor compliant with at least ARM architecture 8.2-a with Cryptographic extensions. On EC2 this is Graviton 2 or later.
```

Membangun gambar sistem operasi

Amazon EKS menyediakan [contoh template Packer](#) yang dapat Anda gunakan untuk membuat gambar sistem operasi yang menyertakan nodeadm dan mengonfigurasinya agar berjalan saat startup host. Proses ini disarankan untuk menghindari penarikan dependensi node hybrid secara individual pada setiap host dan untuk mengotomatiskan proses bootstrap node hybrid. Anda dapat menggunakan contoh template Packer dengan gambar ISO Ubuntu 22.04, Ubuntu 24.04, RHEL 8 atau RHEL 9 dan dapat menampilkan gambar dengan format ini: OVA, Qcow2, atau mentah.

Prasyarat

Sebelum menggunakan contoh template Packer, Anda harus memiliki yang berikut diinstal pada mesin dari tempat Anda menjalankan Packer.

- Packer versi 1.11.0 atau lebih tinggi. Untuk petunjuk tentang menginstal Packer, lihat [Install Packer](#) di dokumentasi Packer.
- Jika membangun OVAs, VMware vSphere plugin 1.4.0 atau lebih tinggi
- Jika membangun Qcow2 atau gambar mentah, plugin QEMU versi 1.x

Tetapkan Variabel Lingkungan

Sebelum menjalankan build Packer, atur variabel lingkungan berikut pada mesin dari tempat Anda menjalankan Packer.

Umum

Variabel lingkungan berikut harus diatur untuk membangun gambar dengan semua sistem operasi dan format output.

Variabel Lingkungan	Tipe	Deskripsi
PKR_SSH_KATA SANDI	String	Packer menggunakan <code>ssh_password</code> variabel <code>ssh_username</code> dan ke SSH ke dalam mesin yang dibuat saat penyediaan. Ini harus sesuai dengan kata sandi yang digunakan untuk membuat pengguna awal dalam file kickstart atau data pengguna OS masing-masing. Default ditetapkan sebagai “builder” atau “ubuntu” tergantung pada OS. Saat mengatur kata sandi Anda, pastikan untuk mengubahnya dalam <code>user-data</code> file yang sesuai <code>ks.cfg</code> atau agar sesuai.
ISO_URL	String	URL ISO yang akan digunakan. Bisa berupa tautan web untuk diunduh dari server, atau jalur absolut ke file lokal
ISO_CHECKSUM	String	Checksum terkait untuk ISO yang disediakan.

Variabel Lingkungan	Tipe	Deskripsi
CREDENTIAL_PROVIDER	String	Penyedia kredensial untuk node hibrida. Nilai yang valid adalah ssm (default) untuk aktivasi hibrida SSM dan iam untuk Peran IAM Di Mana Saja
K8S_VERSION	String	Versi Kubernetes untuk node hybrid (misalnya). 1.31 Untuk versi Kubernetes yang didukung, lihat. the section called “Versi Kubernetes”
NODEADM_ARCH	String	Arsitektur untuk nodeadm install. Pilih amd atau arm.

RHEL

Jika Anda menggunakan RHEL, variabel lingkungan berikut harus diatur.

Variabel Lingkungan	Tipe	Deskripsi
RH_NAMA PENGGUNA	String	Nama pengguna manajer langganan RHEL
RH_KATA SANDI	String	Kata sandi pengelola langganan RHEL
RHEL_VERSION	String	Versi RHEL iso sedang digunakan. Nilai-nilai yang valid adalah 8 atau 9.

Ubuntu

Tidak ada variabel lingkungan khusus Ubuntu yang diperlukan.

vSphere

Jika Anda sedang membangun OVA VMware vSphere, variabel lingkungan berikut harus diatur.

Variabel Lingkungan	Tipe	Deskripsi
VSPHERE_SERVER	String	Alamat server vSphere
VSPHERE_USER	String	Nama pengguna vSphere
VSPHERE_PASSWORD	String	Kata sandi vSphere
VSPHERE_DATACENTER	String	Nama pusat data vSphere
VSPHERE_CLUSTER	String	Nama cluster vSphere
VSPHERE_DATASTORE	String	vSphere nama datastore
VSPHERE_NETWORK	String	nama jaringan vSphere
VSPHERE_OUTPUT_FOLDER	String	folder output vSphere untuk template

QEMU

Variabel Lingkungan	Tipe	Deskripsi
FORMAT PACKER_OUTPUT_	String	Format output untuk pembangun QEMU. Nilai yang valid adalah qcow2 dan raw.

Validasi template

Sebelum menjalankan build, validasi template Anda dengan perintah berikut setelah menyetel variabel lingkungan. Ganti `template.pkr.hcl` jika Anda menggunakan nama yang berbeda untuk template Anda.

```
packer validate template.pkr.hcl
```

Membangun gambar

Bangun gambar Anda dengan perintah berikut dan gunakan `-only` bendera untuk menentukan target dan sistem operasi untuk gambar Anda. Ganti `template.pkr.hcl` jika Anda menggunakan nama yang berbeda untuk template Anda.

vSphere OVA

Note

Jika Anda menggunakan RHEL dengan vSphere, Anda perlu mengonversi file kickstart ke gambar OEMDRV dan meneruskannya sebagai ISO untuk boot. Untuk informasi lebih lanjut, lihat [Packer Readme](#) di EKS Hybrid Nodes GitHub Repository.

Ubuntu 22.04 OVA

```
packer build -only=general-build.vsphere-iso.ubuntu22 template.pkr.hcl
```

Ubuntu 24.04 OVA

```
packer build -only=general-build.vsphere-iso.ubuntu24 template.pkr.hcl
```

RHEL 8 TELUR

```
packer build -only=general-build.vsphere-iso.rhel8 template.pkr.hcl
```

RHEL 9 TELUR

```
packer build -only=general-build.vsphere-iso.rhel9 template.pkr.hcl
```

QEMU

Note

Jika Anda membuat gambar untuk CPU host tertentu yang tidak cocok dengan host pembangun Anda, lihat dokumentasi [QEMU](#) untuk nama yang cocok dengan CPU host Anda dan gunakan `-cpu` tanda dengan nama CPU host saat Anda menjalankan perintah berikut.

Ubuntu 22.04 Qcow2/Mentah

```
packer build -only=general-build.qemu.ubuntu22 template.pkr.hcl
```

Ubuntu 24.04 Qcow2/Mentah

```
packer build -only=general-build.qemu.ubuntu24 template.pkr.hcl
```

RHEL 8 Qcow2/Mentah

```
packer build -only=general-build.qemu.rhel8 template.pkr.hcl
```

RHEL 9 Qcow2/Mentah

```
packer build -only=general-build.qemu.rhel9 template.pkr.hcl
```

Lewati konfigurasi nodeadm melalui data pengguna

Anda dapat meneruskan konfigurasi untuk nodeadm data pengguna Anda melalui cloud-init untuk mengonfigurasi dan secara otomatis menghubungkan node hybrid ke kluster EKS Anda saat startup host. Di bawah ini adalah contoh bagaimana untuk mencapai ini ketika menggunakan VMware vSphere sebagai infrastruktur untuk node hybrid Anda.

1. Instal govc CLI mengikuti instruksi di [govc](#) readme on. GitHub
2. Setelah menjalankan build Packer di bagian sebelumnya dan menyediakan template Anda, Anda dapat mengkloning template Anda untuk membuat beberapa node berbeda menggunakan berikut ini. Anda harus mengkloning template untuk setiap VM baru yang Anda buat yang akan digunakan untuk node hybrid. Ganti variabel dalam perintah di bawah ini dengan nilai untuk lingkungan Anda. Perintah VM_NAME di bawah ini digunakan sebagai Anda NODE_NAME ketika Anda menyuntikkan nama untuk Anda VMs melalui metadata.yaml file Anda.

```
govc vm.clone -vm "/PATH/TO/TEMPLATE" -ds="YOUR_DATASTORE" \
-on=false -template=false -folder=/FOLDER/TO/SAVE/VM "VM_NAME"
```

3. Setelah mengkloning template untuk masing-masing yang baru Anda VMs, buat userdata.yaml dan metadata.yaml untuk Anda VMs. Anda VMs dapat berbagi hal yang sama userdata.yaml metadata.yaml dan dan Anda akan mengisinya berdasarkan per VM dalam langkah-langkah di bawah ini. nodeadmKonfigurasi dibuat dan didefinisikan di write_files bagian Andauserdata.yaml. Contoh di bawah ini menggunakan aktivasi hibrida AWS SSM sebagai

penyedia kredensi lokal untuk node hibrid. Untuk informasi lebih lanjut tentang nodeadm konfigurasi, lihat [the section called “Node hibrida nodeadm”](#).

userdata.yaml:

```
#cloud-config
users:
  - name: # username for login. Use 'builder' for RHEL or 'ubuntu' for Ubuntu.
    passwd: # password to login. Default is 'builder' for RHEL.
    groups: [adm, cdrom, dip, plugdev, lxd, sudo]
    lock-passwd: false
    sudo: ALL=(ALL) NOPASSWD:ALL
    shell: /bin/bash

write_files:
  - path: /usr/local/bin/nodeConfig.yaml
    permissions: '0644'
    content: |
      apiVersion: node.eks.aws/v1alpha1
      kind: NodeConfig
      spec:
        cluster:
          name: # Cluster Name
          region: # AWS region
        hybrid:
          ssm:
            activationCode: # Your ssm activation code
            activationId: # Your ssm activation id

runcmd:
  - /usr/local/bin/nodeadm init -c file:///usr/local/bin/nodeConfig.yaml >> /var/log/
    nodeadm-init.log 2>&1
```

metadata.yaml:

Ciptakan metadata.yaml untuk lingkungan Anda. Simpan format "\$NODE_NAME" variabel dalam file karena ini akan diisi dengan nilai pada langkah berikutnya.

```
instance-id: "$NODE_NAME"
local-hostname: "$NODE_NAME"
network:
  version: 2
```

```
ethernets:
  nics:
    match:
      name: ens*
    dhcp4: yes
```

4. Tambahkan `userdata.yaml` dan `metadata.yaml` file sebagai `gzip+base64` string dengan perintah berikut. Perintah berikut harus dijalankan untuk masing-masing yang VMs Anda buat. Ganti `VM_NAME` dengan nama VM yang Anda perbarui.

```
export NODE_NAME="VM_NAME"
export USER_DATA=$(gzip -c9 <userdata.yaml | base64)

govc vm.change -dc="YOUR_DATASTORE" -vm "$NODE_NAME" -e
  guestinfo.userdata="${USER_DATA}"
govc vm.change -dc="YOUR_DATASTORE" -vm "$NODE_NAME" -e
  guestinfo.userdata.encoding=gzip+base64

envsubst '$NODE_NAME' < metadata.yaml > metadata.yaml.tmp
export METADATA=$(gzip -c9 <metadata.yaml.tmp | base64)

govc vm.change -dc="YOUR_DATASTORE" -vm "$NODE_NAME" -e
  guestinfo.metadata="${METADATA}"
govc vm.change -dc="YOUR_DATASTORE" -vm "$NODE_NAME" -e
  guestinfo.metadata.encoding=gzip+base64
```

5. Nyalakan yang baru Anda VMs, yang akan secara otomatis terhubung ke cluster EKS yang Anda konfigurasi.

```
govc vm.power -on "${NODE_NAME}"
```

Siapkan kredensial untuk node hybrid

Amazon EKS Hybrid Nodes menggunakan kredensial IAM sementara yang disediakan oleh aktivasi hybrid AWS SSM atau AWS Peran IAM Anywhere untuk mengautentikasi dengan kluster Amazon EKS. Anda harus menggunakan aktivasi hibrida AWS SSM atau Peran AWS IAM Di Mana Saja dengan Amazon EKS Hybrid Nodes CLI (). `nodeadm` Anda tidak boleh menggunakan aktivasi hibrida AWS SSM dan Peran AWS IAM Di Mana Saja. Kami menyarankan Anda menggunakan aktivasi hibrida AWS SSM jika Anda tidak memiliki Infrastruktur Kunci Publik (PKI) dengan Otoritas Sertifikat

(CA) dan sertifikat untuk lingkungan lokal Anda. Jika Anda memiliki PKI dan sertifikat yang ada di lokasi, gunakan Peran AWS IAM Di Mana Saja.

Peran IAM Node Hibrida

Sebelum Anda dapat menghubungkan node hybrid ke cluster Amazon EKS Anda, Anda harus membuat peran IAM yang akan digunakan dengan aktivasi hibrida AWS SSM atau Peran AWS IAM Anywhere untuk kredensial node hybrid Anda. Setelah pembuatan klaster, Anda akan menggunakan peran ini dengan entri akses Amazon EKS atau `aws-auth ConfigMap` entri untuk memetakan peran IAM ke Kubernetes Role-Based Access Control (RBAC). Untuk informasi lebih lanjut tentang mengaitkan peran IAM Hybrid Nodes dengan Kubernetes RBAC, lihat [the section called “Siapkan akses cluster”](#)

Peran IAM Hybrid Nodes harus memiliki izin berikut.

- Izin `nodeadm` untuk menggunakan `eks:DescribeCluster` tindakan untuk mengumpulkan informasi tentang cluster yang digunakan untuk menghubungkan node hibrida ke cluster. Jika Anda tidak mengaktifkan `eks:DescribeCluster` action, maka Anda harus meneruskan endpoint Kubernetes API, cluster CA bundle, dan service IPv4 CIDR dalam konfigurasi node yang Anda berikan ketika Anda menjalankan `init. nodeadm nodeadm`
- [Izin untuk kubelet untuk menggunakan image container dari Amazon Elastic Container Registry \(Amazon ECR\) seperti yang didefinisikan dalam kebijakan Amazon. EC2 ContainerRegistryPullOnly](#)
- [Jika menggunakan AWS SSM, izin untuk nodeadm init untuk menggunakan aktivasi hibrida AWS SSM seperti yang didefinisikan dalam kebijakan/.html. aws-managed-policy latest/reference/AmazonSSMManagedInstanceCore](#)
- Jika menggunakan AWS SSM, izin untuk menggunakan `ssm:DeregisterManagedInstance` tindakan dan `ssm:DescribeInstanceInformation` tindakan `nodeadm uninstall` untuk membatalkan pendaftaran instance.
- (Opsional) Izin untuk Amazon EKS Pod Identity Agent untuk menggunakan `eks-auth:AssumeRoleForPodIdentity` action tersebut guna mengambil kredensial Pod.

Siapkan AWS aktivasi hibrida SSM

Sebelum menyiapkan aktivasi hybrid AWS SSM, Anda harus memiliki peran IAM Hybrid Nodes yang dibuat dan dikonfigurasi. Untuk informasi selengkapnya, lihat [the section called “Buat peran IAM Hybrid Nodes”](#). Ikuti petunjuk di [Buat aktivasi hybrid untuk mendaftarkan node dengan Systems](#)

[Manager](#) di Panduan Pengguna AWS Systems Manager untuk membuat aktivasi hibrida AWS SSM untuk node hybrid Anda. Kode Aktivasi dan ID yang Anda terima digunakan nodeadm saat Anda mendaftarkan host Anda sebagai node hibrida dengan kluster Amazon EKS Anda. Anda dapat kembali ke langkah ini di lain waktu setelah Anda membuat dan menyiapkan cluster Amazon EKS Anda untuk node hibrida.

 Important

Systems Manager akan segera mengembalikan Kode Aktivasi dan ID ke konsol atau jendela perintah, tergantung pada bagaimana Anda membuat aktivasi. Salin informasi ini dan simpan di tempat yang aman. Jika Anda menavigasi keluar dari konsol atau menutup jendela perintah, Anda mungkin kehilangan informasi ini. Jika Anda kehilangannya, Anda harus membuat aktivasi baru.

Secara default, aktivasi hibrida AWS SSM aktif selama 24 jam. Anda juga dapat menentukan --expiration-date kapan Anda membuat aktivasi hibrida dalam format stempel waktu, seperti. 2024-08-01T00:00:00 Saat Anda menggunakan AWS SSM sebagai penyedia kredensi Anda, nama node untuk node hibrida Anda tidak dapat dikonfigurasi, dan dibuat secara otomatis oleh SSM. AWS Anda dapat melihat dan mengelola Instans Terkelola AWS SSM di konsol AWS Systems Manager di bawah Fleet Manager. Anda dapat mendaftarkan hingga 1.000 [node standar yang diaktifkan hibrida](#) per akun per AWS Wilayah tanpa biaya tambahan. Namun, mendaftarkan lebih dari 1.000 node hybrid mengharuskan Anda mengaktifkan tingkat instance lanjutan. Ada biaya untuk menggunakan tingkat instans lanjutan yang tidak termasuk dalam harga [Amazon EKS Hybrid Nodes](#). Untuk informasi selengkapnya, lihat [Harga AWS Systems Manager](#).

Lihat contoh di bawah ini untuk cara membuat aktivasi hybrid AWS SSM dengan peran IAM Hybrid Nodes Anda. Saat Anda menggunakan aktivasi hibrida AWS SSM untuk kredensial node hybrid Anda, nama-nama node hibrida Anda akan memiliki format mi-012345678abcdefgh dan kredensial sementara yang disediakan oleh SSM berlaku selama 1 jam. AWS Anda tidak dapat mengubah nama node atau durasi kredensi saat menggunakan AWS SSM sebagai penyedia kredensi Anda. Kredensial sementara secara otomatis diputar oleh AWS SSM dan rotasi tidak memengaruhi status node atau aplikasi Anda.

Kami menyarankan Anda menggunakan satu aktivasi hibrida AWS SSM per kluster EKS untuk cakupan ssm:DeregisterManagedInstance izin AWS SSM dari peran IAM Hybrid Nodes agar hanya dapat membatalkan pendaftaran instance yang terkait dengan aktivasi hibrida SSM Anda. AWS Dalam contoh di halaman ini, tag dengan ARN cluster EKS digunakan, yang dapat digunakan

untuk memetakan aktivasi hibrida AWS SSM Anda ke cluster EKS. Sebagai alternatif, Anda dapat menggunakan tag dan metode pilihan Anda untuk melingkupi izin AWS SSM berdasarkan batasan dan persyaratan izin Anda. `REGISTRATION_LIMIT` opsi dalam perintah di bawah ini adalah bilangan bulat yang digunakan untuk membatasi jumlah mesin yang dapat menggunakan aktivasi hibrida AWS SSM (misalnya) 10

```
aws ssm create-activation \
  --region AWS_REGION \
  --default-instance-name eks-hybrid-nodes \
  --description "Activation for EKS hybrid nodes" \
  --iam-role AmazonEKSHybridNodesRole \
  --tags Key=EKSclusterARN,Value=arn:aws: eks:AWS_REGION:AWS_ACCOUNT_ID:cluster/
  CLUSTER_NAME \
  --registration-limit REGISTRATION_LIMIT
```

Tinjau instruksi tentang [Buat aktivasi hybrid untuk mendaftarkan node dengan Systems Manager](#) untuk informasi lebih lanjut tentang pengaturan konfigurasi yang tersedia untuk aktivasi hibrida AWS SSM.

Siapkan Peran AWS IAM Di Mana Saja

Ikuti petunjuk di [Memulai Peran IAM Di Mana Saja](#) di Panduan Pengguna IAM Roles Anywhere untuk menyiapkan jangkar kepercayaan dan profil yang akan Anda gunakan untuk kredensial IAM sementara untuk peran IAM Hybrid Nodes Anda. Saat Anda membuat profil, Anda dapat membuatnya tanpa menambahkan peran apa pun. Anda dapat membuat profil ini, kembali ke langkah-langkah ini untuk membuat peran IAM Hybrid Nodes Anda, dan kemudian menambahkan peran Anda ke profil Anda setelah dibuat. Anda dapat menggunakan AWS CloudFormation langkah-langkah selanjutnya di halaman ini untuk menyelesaikan penyiapan IAM Roles Anywhere Anda untuk node hybrid.

Saat Anda menambahkan peran IAM Hybrid Nodes ke profil Anda, pilih Terima nama sesi peran kustom di panel Nama sesi peran kustom di bagian bawah halaman Edit profil di konsol AWS IAM Roles Anywhere. Ini sesuai dengan bidang `acceptRoleSessionName` `CreateProfile` API. Ini memungkinkan Anda untuk memberikan nama node khusus untuk node hybrid Anda dalam konfigurasi yang Anda berikan `nodeadm` selama proses bootstrap. Meneruskan nama node kustom selama `nodeadm init` proses diperlukan. Anda dapat memperbarui profil Anda untuk menerima nama sesi peran khusus setelah membuat profil Anda.

Anda dapat mengonfigurasi durasi validitas kredensi dengan AWS IAM Roles Anywhere melalui bidang `DurationSeconds` pada profil IAM Roles [Anywhere](#) Anda. AWS Durasi default adalah 1 jam dengan maksimal 12 jam. `MaxSessionDuration` pengaturan pada peran IAM Hybrid Nodes Anda

harus lebih besar dari `durationSeconds` pengaturan pada profil AWS IAM Roles Anywhere Anda. Untuk informasi selengkapnya `MaxSessionDuration`, lihat [dokumentasi UpdateRole API](#).

Sertifikat dan kunci per mesin yang Anda hasilkan dari otoritas sertifikat (CA) Anda harus ditempatkan di `/etc/iam/pki` direktori pada setiap node hibrida dengan nama file `server.pem` untuk sertifikat dan `server.key` untuk kunci.

Buat peran IAM Hybrid Nodes

Untuk menjalankan langkah-langkah di bagian ini, prinsipal IAM yang menggunakan AWS konsol atau AWS CLI harus memiliki izin berikut.

- `iam:CreatePolicy`
- `iam:CreateRole`
- `iam:AttachRolePolicy`
- Jika menggunakan Peran AWS IAM Di Mana Saja
 - `rolesanywhere:CreateTrustAnchor`
 - `rolesanywhere:CreateProfile`
 - `iam:PassRole`

AWS CloudFormation

Instal dan konfigurasi AWS CLI, jika Anda belum melakukannya. Lihat [Menginstal atau memperbarui ke versi terakhir AWS CLI](#).

Langkah-langkah untuk aktivasi hibrida AWS SSM

CloudFormation Tumpukan membuat Peran IAM Hybrid Nodes dengan izin yang diuraikan di atas. CloudFormation Template tidak membuat aktivasi hibrida AWS SSM.

1. Unduh CloudFormation template AWS SSM untuk node hybrid:

```
curl -OL 'https://raw.githubusercontent.com/aws/eks-hybrid/refs/heads/main/example/hybrid-ssm-cfn.yaml'
```

2. Buat `cfn-ssm-parameters.json` dengan opsi berikut:

- a. Ganti `ROLE_NAME` dengan nama untuk peran IAM Hybrid Nodes Anda. Secara default, CloudFormation template digunakan `AmazonEKSHybridNodesRole` sebagai nama peran yang dibuatnya jika Anda tidak menentukan nama.

- b. Ganti TAG_KEY dengan kunci tag sumber daya AWS SSM yang Anda gunakan saat membuat aktivasi hibrida AWS SSM Anda. Kombinasi kunci tag dan nilai tag digunakan dalam kondisi untuk hanya mengizinkan peran IAM Hybrid Nodes `ssm:DeregisterManagedInstance` untuk membatalkan pendaftaran instance terkelola AWS SSM yang terkait dengan aktivasi hibrida SSM Anda. AWS Dalam CloudFormation template, TAG_KEY default ke `EKSClusterARN`.
- c. Ganti TAG_VALUE dengan nilai tag sumber daya AWS SSM yang Anda gunakan saat membuat aktivasi hibrida AWS SSM Anda. Kombinasi kunci tag dan nilai tag digunakan dalam kondisi untuk hanya mengizinkan peran IAM Hybrid Nodes `ssm:DeregisterManagedInstance` untuk membatalkan pendaftaran instance terkelola AWS SSM yang terkait dengan aktivasi hibrida SSM Anda. AWS Jika Anda menggunakan default TAG_KEY dari `EKSClusterARN`, maka berikan ARN cluster EKS Anda sebagai ARN. TAG_VALUE Kluster EKS ARNs memiliki format `arn:aws:eks:AWS_REGION:AWS_ACCOUNT_ID:cluster/CLUSTER_NAME`.

```
{
  "Parameters": {
    "RoleName": "ROLE_NAME",
    "SSMDeregisterConditionTagKey": "TAG_KEY",
    "SSMDeregisterConditionTagValue": "TAG_VALUE"
  }
}
```

3. Menyebarkan CloudFormation tumpukan. Ganti STACK_NAME dengan nama Anda untuk CloudFormation tumpukan.

```
aws cloudformation deploy \
  --stack-name STACK_NAME \
  --template-file hybrid-ssm-cfn.yaml \
  --parameter-overrides file://cfn-ssm-parameters.json \
  --capabilities CAPABILITY_NAMED_IAM
```

Langkah-langkah untuk Peran AWS IAM Di Mana Saja

CloudFormation Tumpukan membuat jangkar kepercayaan AWS IAM Roles Anywhere dengan certificate authority (CA) yang Anda konfigurasikan, membuat profil AWS IAM Roles Anywhere, dan membuat peran IAM Hybrid Nodes dengan izin yang diuraikan sebelumnya.

1. Untuk mengatur otoritas sertifikat (CA)

- a. Untuk menggunakan sumber daya CA AWS Pribadi, buka [konsol AWS Private Certificate Authority](#). Ikuti petunjuk di [Panduan Pengguna CA AWS Pribadi](#).
 - b. Untuk menggunakan CA eksternal, ikuti instruksi yang diberikan oleh CA. Anda memberikan badan sertifikat di langkah selanjutnya.
 - c. Sertifikat yang dikeluarkan dari publik CAs tidak dapat digunakan sebagai jangkar kepercayaan.
2. Unduh CloudFormation template AWS IAM Roles Anywhere untuk node hybrid

```
curl -OL 'https://raw.githubusercontent.com/aws/eks-hybrid/refs/heads/main/example/hybrid-ira-cfn.yaml'
```

3. Buat `cfn-iamra-parameters.json` dengan opsi berikut:
 - a. Ganti `ROLE_NAME` dengan nama untuk peran IAM Hybrid Nodes Anda. Secara default, CloudFormation template digunakan `AmazonEKSHybridNodesRole` sebagai nama peran yang dibuatnya jika Anda tidak menentukan nama.
 - b. Ganti `CERT_ATTRIBUTE` dengan atribut sertifikat per-mesin yang secara unik mengidentifikasi host Anda. Atribut certificate yang Anda gunakan harus cocok dengan `nodeName` yang Anda gunakan untuk `nodeadm` konfigurasi saat Anda menghubungkan node hybrid ke cluster Anda. Untuk informasi selengkapnya, lihat [the section called "Node hibrida nodeadm"](#). Secara default, CloudFormation template menggunakan `${aws:PrincipalTag/x509Subject/CN}` sebagai `CERT_ATTRIBUTE`, yang sesuai dengan bidang CN sertifikat per-mesin Anda. Anda bisa lolos `$(aws:PrincipalTag/x509SAN/Name/CN)` sebagai milik Anda `CERT_ATTRIBUTE`.
 - c. Ganti `CA_CERT_BODY` dengan badan sertifikat CA Anda tanpa jeda baris. `CA_CERT_BODY` harus dalam format Privacy Enhanced Mail (PEM). Jika Anda memiliki sertifikat CA dalam format PEM, hapus jeda baris dan MULAI SERTIFIKAT dan AKHIR SERTIFIKAT sebelum menempatkan badan sertifikat CA di `cfn-iamra-parameters.json` file Anda.

```
{
  "Parameters": {
    "RoleName": "ROLE_NAME",
    "CertAttributeTrustPolicy": "CERT_ATTRIBUTE",
    "CABundleCert": "CA_CERT_BODY"
  }
}
```

4. Menyebarkan CloudFormation template. Ganti `STACK_NAME` dengan nama Anda untuk CloudFormation tumpukan.

```
aws cloudformation deploy \  
  --stack-name STACK_NAME \  
  --template-file hybrid-ira-cfn.yaml \  
  --parameter-overrides file://cfn-iamra-parameters.json \  
  --capabilities CAPABILITY_NAMED_IAM
```

AWS CLI

Instal dan konfigurasi AWS CLI, jika Anda belum melakukannya. Lihat [Menginstal atau memperbarui ke versi terakhir AWS CLI](#).

Buat EKS Jelaskan Kebijakan Cluster

1. Buat file bernama `eks-describe-cluster-policy.json` dengan konten berikut:

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "eks:DescribeCluster"  
      ],  
      "Resource": "*"   
    }  
  ]  
}
```

2. Buat kebijakan dengan perintah berikut:

```
aws iam create-policy \  
  --policy-name EKSDescribeClusterPolicy \  
  --policy-document file://eks-describe-cluster-policy.json
```

Langkah-langkah untuk aktivasi hibrida AWS SSM

1. Buat file bernama `eks-hybrid-ssm-policy.json` dengan isi berikut ini. Kebijakan tersebut memberikan izin untuk dua tindakan `ssm:DescribeInstanceInformation` dan `ssm:DeregisterManagedInstance`. Kebijakan ini membatasi

`ssm:DeregisterManagedInstance` izin untuk instans terkelola AWS SSM yang terkait dengan aktivasi hibrida AWS SSM Anda berdasarkan tag sumber daya yang Anda tentukan dalam kebijakan kepercayaan Anda.

- a. Ganti `AWS_REGION` dengan AWS Region untuk aktivasi hybrid AWS SSM Anda.
- b. Ganti `AWS_ACCOUNT_ID` dengan ID AWS akun Anda.
- c. Ganti `TAG_KEY` dengan kunci tag sumber daya AWS SSM yang Anda gunakan saat membuat aktivasi hibrida AWS SSM Anda. Kombinasi kunci tag dan nilai tag digunakan dalam kondisi untuk hanya mengizinkan peran IAM Hybrid Nodes `ssm:DeregisterManagedInstance` untuk membatalkan pendaftaran instance terkelola AWS SSM yang terkait dengan aktivasi hibrida SSM Anda. AWS Dalam CloudFormation template, `TAG_KEY` default ke `EKSclusterARN`
- d. Ganti `TAG_VALUE` dengan nilai tag sumber daya AWS SSM yang Anda gunakan saat membuat aktivasi hibrida AWS SSM Anda. Kombinasi kunci tag dan nilai tag digunakan dalam kondisi untuk hanya mengizinkan peran IAM Hybrid Nodes `ssm:DeregisterManagedInstance` untuk membatalkan pendaftaran instance terkelola AWS SSM yang terkait dengan aktivasi hibrida SSM Anda. AWS Jika Anda menggunakan default `TAG_KEY` dari `EKSclusterARN`, maka berikan ARN cluster EKS Anda sebagai ARN. `TAG_VALUE` Kluster EKS ARNs memiliki format `arn:aws: eks:AWS_REGION:AWS_ACCOUNT_ID:cluster/CLUSTER_NAME`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ssm:DescribeInstanceInformation",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "ssm:DeregisterManagedInstance",
      "Resource": "arn:aws: ssm:AWS_REGION:AWS_ACCOUNT_ID:managed-instance/
*",
      "Condition": {
        "StringEquals": {
          "ssm:resourceTag/TAG_KEY": "TAG_VALUE"
        }
      }
    }
  ]
}
```

```
}
```

2. Buat kebijakan dengan perintah berikut

```
aws iam create-policy \  
  --policy-name EKSHybridSSMPolicy \  
  --policy-document file://eks-hybrid-ssm-policy.json
```

3. Buat file bernama `eks-hybrid-ssm-trust.json`. Ganti `AWS_REGION` dengan AWS Wilayah aktivasi hibrida AWS SSM Anda dan `AWS_ACCOUNT_ID` dengan ID AWS akun Anda.

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "",  
      "Effect": "Allow",  
      "Principal": {  
        "Service": "ssm.amazonaws.com"  
      },  
      "Action": "sts:AssumeRole",  
      "Condition": {  
        "StringEquals": {  
          "aws:SourceAccount": "AWS_ACCOUNT_ID"  
        },  
        "ArnEquals": {  
          "aws:SourceArn": "arn:aws:ssm:AWS_REGION:AWS_ACCOUNT_ID:*"  
        }  
      }  
    }  
  ]  
}
```

4. Buat peran dengan perintah berikut.

```
aws iam create-role \  
  --role-name AmazonEKSHybridNodesRole \  
  --assume-role-policy-document file://eks-hybrid-ssm-trust.json
```

5. Lampirkan `EKSDescribeClusterPolicy` dan yang `EKSHybridSSMPolicy` Anda buat di langkah sebelumnya. Ganti `AWS_ACCOUNT_ID` dengan ID AWS akun Anda.

```
aws iam attach-role-policy \  
  --role-name AmazonEKSHybridNodesRole
```

```
--role-name AmazonEKSHybridNodesRole \
--policy-arn arn:aws:iam::AWS_ACCOUNT_ID:policy/EKSDescribeClusterPolicy
```

```
aws iam attach-role-policy \
  --role-name AmazonEKSHybridNodesRole \
  --policy-arn arn:aws:iam::AWS_ACCOUNT_ID:policy/EKSHybridSSMPolicy
```

6. Lampirkan AmazonEC2ContainerRegistryPullOnly dan kebijakan yang AmazonSSMManagedInstanceCore AWS dikelola.

```
aws iam attach-role-policy \
  --role-name AmazonEKSHybridNodesRole \
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryPullOnly
```

```
aws iam attach-role-policy \
  --role-name AmazonEKSHybridNodesRole \
  --policy-arn arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
```

Langkah-langkah untuk Peran AWS IAM Di Mana Saja

Untuk menggunakan Peran AWS IAM Di Mana Saja, Anda harus menyiapkan jangkar kepercayaan AWS IAM Roles Anywhere sebelum membuat Peran IAM Hybrid Nodes. Lihat [the section called “Siapkan Peran AWS IAM Di Mana Saja”](#) untuk instruksi.

1. Buat file bernama `eks-hybrid-iamra-trust.json`. Ganti `TRUST_ANCHOR` ARN dengan ARN dari jangkar kepercayaan yang Anda buat di langkah-langkahnya. [the section called “Siapkan Peran AWS IAM Di Mana Saja”](#) Kondisi dalam kebijakan kepercayaan ini membatasi kemampuan IAM Roles Anywhere untuk mengambil peran AWS IAM Hybrid Nodes untuk bertukar kredensial IAM sementara hanya jika nama sesi peran cocok dengan CN dalam sertifikat x509 yang diinstal pada node hybrid Anda. Anda dapat menggunakan atribut sertifikat lain untuk mengidentifikasi node Anda secara unik. Atribut sertifikat yang Anda gunakan dalam kebijakan kepercayaan harus sesuai dengan yang `nodeName` Anda tetapkan dalam `nodeadm` konfigurasi Anda. Untuk informasi selengkapnya, lihat [the section called “Node hibrida nodeadm”](#).

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```

        "Effect": "Allow",
        "Principal": {
            "Service": "rolesanywhere.amazonaws.com"
        },
        "Action": [
            "sts:TagSession",
            "sts:SetSourceIdentity"
        ],
        "Condition": {
            "ArnEquals": {
                "aws:SourceArn": "TRUST_ANCHOR_ARN"
            }
        }
    },
    {
        "Effect": "Allow",
        "Principal": {
            "Service": "rolesanywhere.amazonaws.com"
        },
        "Action": "sts:AssumeRole",
        "Condition": {
            "StringEquals": {
                "sts:RoleSessionName": "${aws:PrincipalTag/x509Subject/CN}"
            },
            "ArnEquals": {
                "aws:SourceArn": "TRUST_ANCHOR_ARN"
            }
        }
    }
]
}

```

2. Buat peran dengan perintah berikut.

```

aws iam create-role \
  --role-name AmazonEKSHybridNodesRole \
  --assume-role-policy-document file://eks-hybrid-iamra-trust.json

```

3. Lampirkan yang `EKSDescribeClusterPolicy` Anda buat di langkah sebelumnya. Ganti `AWS_ACCOUNT_ID` dengan ID AWS akun Anda.

```

aws iam attach-role-policy \
  --role-name AmazonEKSHybridNodesRole \

```

```
--policy-arn arn:aws:iam::AWS_ACCOUNT_ID:policy/EKSDescribeClusterPolicy
```

4. Lampirkan kebijakan AmazonEC2ContainerRegistryPullOnly AWS terkelola

```
aws iam attach-role-policy \  
  --role-name AmazonEKSHybridNodesRole \  
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryPullOnly
```

AWS Management Console

Buat EKS Jelaskan Kebijakan Cluster

1. Buka konsol [Amazon IAM](#)
2. Di panel navigasi di sebelah kiri, pilih Kebijakan.
3. Pada halaman Kebijakan, pilih Buat kebijakan.
4. Pada halaman Tentukan izin, di panel Select a Service, pilih EKS.
 - a. Filter tindakan untuk DescribeCluster dan pilih tindakan DescribeClusterBaca.
 - b. Pilih Berikutnya.
5. Pada halaman Review dan buat
 - a. Masukkan nama Kebijakan untuk kebijakan Anda seperti EKSDescribeClusterPolicy.
 - b. Pilih Buat kebijakan.

Langkah-langkah untuk aktivasi hibrida AWS SSM

1. Buka konsol [Amazon IAM](#)
2. Di panel navigasi di sebelah kiri, pilih Kebijakan.
3. Pada halaman Kebijakan, pilih Buat kebijakan.
4. Pada halaman Tentukan izin, di navigasi kanan atas editor Kebijakan, pilih JSON. Tempel cuplikan berikut. Ganti AWS_REGION dengan AWS Wilayah aktivasi hibrida AWS SSM Anda dan ganti AWS_ACCOUNT_ID dengan ID AWS akun Anda. Ganti TAG_KEY dan TAG_VALUE dengan kunci tag sumber daya AWS SSM yang Anda gunakan saat membuat aktivasi hibrida AWS SSM Anda.

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {
```

```

        "Effect": "Allow",
        "Action": "ssm:DescribeInstanceInformation",
        "Resource": "*"
    },
    {
        "Effect": "Allow",
        "Action": "ssm:DeregisterManagedInstance",
        "Resource": "arn:aws:ssm:AWS_REGION:AWS_ACCOUNT_ID:managed-instance/*",
        "Condition": {
            "StringEquals": {
                "ssm:resourceTag/TAG_KEY": "TAG_VALUE"
            }
        }
    }
]
}

```

- a. Pilih Berikutnya.
5. Pada halaman Review dan Create.
 - a. Masukkan nama Kebijakan untuk kebijakan Anda seperti EKSHybridSSMPolicy
 - b. Pilih Buat Kebijakan.
6. Di panel navigasi sebelah kiri, pilih Peran.
7. Pada halaman Peran, pilih Buat peran.
8. Pada halaman Pilih entitas tepercaya, lakukan hal berikut:
 - a. Di bagian Jenis entitas tepercaya, pilih Kebijakan kepercayaan khusus. Tempelkan berikut ini ke editor kebijakan kepercayaan kustom. Ganti AWS_REGION dengan AWS Wilayah aktivasi hibrida AWS SSM Anda dan AWS_ACCOUNT_ID dengan ID AWS akun Anda.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "ssm.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {

```



```

        "aws:SourceAccount":"AWS_ACCOUNT_ID"
      },
      "ArnEquals":{
        "aws:SourceArn":"arn:aws:ssm:AWS_REGION:AWS_ACCOUNT_ID:*"
      }
    }
  }
]
}

```

b. Pilih Berikutnya.

9. Pada halaman Tambahkan izin, lampirkan kebijakan khusus atau lakukan hal berikut:

- Di kotak Filter kebijakan, masukkan `EKSDescribeClusterPolicy`, atau nama kebijakan yang Anda buat di atas. Pilih kotak centang di sebelah kiri nama kebijakan Anda di hasil penelusuran.
- Di kotak Filter kebijakan, masukkan `EKSHybridSSMPolicy`, atau nama kebijakan yang Anda buat di atas. Pilih kotak centang di sebelah kiri nama kebijakan Anda di hasil penelusuran.
- Di dalam kotak Filter kebijakan, masukkan `AmazonEC2ContainerRegistryPullOnly`. Pilih kotak centang di sebelah kiri `AmazonEC2ContainerRegistryPullOnly` dalam hasil pencarian.
- Di dalam kotak Filter kebijakan, masukkan `AmazonSSMManagedInstanceCore`. Pilih kotak centang di sebelah kiri `AmazonSSMManagedInstanceCore` dalam hasil pencarian.
- Pilih Berikutnya.

10 Pada halaman Nama, tinjau, dan buat, lakukan hal berikut:

- Untuk nama Peran, masukkan nama unik untuk peran Anda, seperti `AmazonEKSHybridNodesRole`.
- Untuk Deskripsi, ganti teks saat ini dengan teks deskriptif seperti `Amazon EKS - Hybrid Nodes role`.
- Pilih Buat peran.

Langkah-langkah untuk Peran AWS IAM Di Mana Saja

Untuk menggunakan Peran AWS IAM Di Mana Saja, Anda harus menyiapkan jangkar kepercayaan AWS IAM Roles Anywhere sebelum membuat Peran IAM Hybrid Nodes. Lihat [the section called “Siapkan Peran AWS IAM Di Mana Saja”](#) untuk instruksi.

- Buka konsol [Amazon IAM](#)
- Di panel navigasi sebelah kiri, pilih Peran.

3. Pada halaman Peran, pilih Buat peran.
4. Pada halaman Pilih entitas tepercaya, lakukan hal berikut:
 - a. Di bagian Jenis entitas tepercaya, pilih Kebijakan kepercayaan khusus. Tempelkan berikut ini ke editor kebijakan kepercayaan kustom. Ganti TRUST_ANCHOR ARN dengan ARN dari jangkar kepercayaan yang Anda buat di langkah-langkahnya. [the section called “Siapkan Peran AWS IAM Di Mana Saja”](#) Kondisi dalam kebijakan kepercayaan ini membatasi kemampuan IAM Roles Anywhere untuk mengambil peran AWS IAM Hybrid Nodes untuk bertukar kredensial IAM sementara hanya jika nama sesi peran cocok dengan CN dalam sertifikat x509 yang diinstal pada node hybrid Anda. Anda dapat menggunakan atribut sertifikat lain untuk mengidentifikasi node Anda secara unik. Atribut sertifikat yang Anda gunakan dalam kebijakan trust harus sesuai dengan NodeName yang Anda tetapkan dalam konfigurasi nodeadm Anda. Untuk informasi selengkapnya, lihat [the section called “Node hibrida nodeadm”](#).

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "rolesanywhere.amazonaws.com"
      },
      "Action": [
        "sts:TagSession",
        "sts:SetSourceIdentity"
      ],
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "TRUST_ANCHOR_ARN"
        }
      }
    },
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "rolesanywhere.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "sts:RoleSessionName": "${aws:PrincipalTag/x509Subject/CN}"
        }
      }
    }
  ]
}
```

```

        "ArnEquals": {
            "aws:SourceArn": "TRUST_ANCHOR_ARN"
        }
    }
}
]
}

```

- b. Pilih Berikutnya.
5. Pada halaman Tambahkan izin, lampirkan kebijakan khusus atau lakukan hal berikut:
 - a. Di kotak Filter kebijakan, masukkan `EKSDescribeClusterPolicy`, atau nama kebijakan yang Anda buat di atas. Pilih kotak centang di sebelah kiri nama kebijakan Anda di hasil penelusuran.
 - b. Di dalam kotak Filter kebijakan, masukkan `AmazonEC2ContainerRegistryPullOnly`. Pilih kotak centang di sebelah kiri `AmazonEC2ContainerRegistryPullOnly` dalam hasil pencarian.
 - c. Pilih Berikutnya.
6. Pada halaman Nama, tinjau, dan buat, lakukan hal berikut:
 - a. Untuk nama Peran, masukkan nama unik untuk peran Anda, seperti `AmazonEKSHybridNodesRole`.
 - b. Untuk Deskripsi, ganti teks saat ini dengan teks deskriptif seperti `Amazon EKS - Hybrid Nodes role`.
 - c. Pilih Buat peran.

Buat kluster Amazon EKS dengan node hybrid

Topik ini memberikan ikhtisar opsi yang tersedia dan menjelaskan apa yang harus dipertimbangkan saat Anda membuat kluster Amazon EKS berkemampuan node hibrida. EKS Hybrid Nodes memiliki dukungan [versi Kubernetes](#) yang sama dengan cluster Amazon EKS dengan node cloud, termasuk dukungan standar dan diperpanjang.

Jika Anda tidak berencana menggunakan EKS Hybrid Nodes, lihat dokumentasi kluster Amazon EKS utama yang membuat di [the section called “Membuat kluster”](#).

Prasyarat

- [the section called “Prasyarat”](#) Selesai. Sebelum Anda membuat kluster berkemampuan node hibrid, node lokal dan pod Anda harus diidentifikasi secara opsional CIDRs, VPC dan subnet Anda dibuat sesuai dengan persyaratan EKS, dan persyaratan node hybrid, dan grup keamanan Anda dengan

aturan masuk untuk pod lokal dan opsional Anda. CIDRs Untuk informasi lebih lanjut tentang prasyarat ini, lihat. [the section called “Siapkan jaringan”](#)

- Versi terbaru dari AWS Command Line Interface (AWS CLI) diinstal dan dikonfigurasi pada perangkat Anda. Untuk memeriksa versi Anda saat ini, gunakan `aws --version`. Package manager seperti yum, apt-get, atau Homebrew untuk macOS seringkali merupakan beberapa versi di belakang versi terbaru CLI. AWS Untuk menginstal versi terbaru, lihat [Menginstal atau memperbarui ke versi terakhir CLI dan Mengkonfigurasi pengaturan untuk AWS CLI di Panduan Pengguna](#) Antarmuka Baris AWS Perintah.
- [Prinsipal IAM](#) dengan izin untuk membuat peran IAM dan melampirkan kebijakan, serta membuat dan mendeskripsikan kluster EKS

Pertimbangan

- Cluster Anda harus menggunakan salah satu API atau API_AND_CONFIG_MAP untuk mode otentikasi cluster.
- Cluster Anda harus menggunakan keluarga IPv4 alamat.
- Cluster Anda harus menggunakan konektivitas titik akhir kluster Publik atau Pribadi. Cluster Anda tidak dapat menggunakan konektivitas endpoint kluster “Publik dan Pribadi”, karena endpoint server Amazon EKS Kubernetes API akan menyelesaikan ke publik IPs untuk node hybrid yang berjalan di luar VPC Anda.
- Otentikasi OIDC didukung untuk kluster EKS dengan node hybrid.
- Anda dapat menambahkan, mengubah, atau menghapus konfigurasi node hybrid dari cluster yang ada. Untuk informasi selengkapnya, lihat [the section called “Cluster yang ada”](#).

Langkah 1: Buat peran IAM cluster

Jika Anda sudah memiliki peran IAM cluster, atau Anda akan membuat cluster Anda dengan `eksctl` atau AWS CloudFormation, maka Anda dapat melewati langkah ini. Secara default, `eksctl` dan AWS CloudFormation template membuat peran IAM cluster untuk Anda.

1. Jalankan perintah berikut untuk membuat file JSON kebijakan kepercayaan IAM.

```
cat >eks-cluster-role-trust-policy.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```

    "Effect": "Allow",
    "Principal": {
        "Service": "eks.amazonaws.com"
    },
    "Action": "sts:AssumeRole"
}
]
}
EOF

```

2. Buat peran IAM cluster Amazon EKS. Jika perlu, kata pengantar eks-cluster-role-trust-policy.json dengan jalur di komputer Anda tempat Anda menulis file di langkah sebelumnya. Perintah tersebut mengaitkan kebijakan kepercayaan yang Anda buat pada langkah sebelumnya dengan peran tersebut. Untuk membuat peran IAM, [prinsipal IAM](#) yang membuat peran harus diberi iam:CreateRole tindakan (izin).

```

aws iam create-role \
  --role-name myAmazonEKSClusterRole \
  --assume-role-policy-document file://"eks-cluster-role-trust-policy.json"

```

3. Anda dapat menetapkan kebijakan terkelola Amazon EKS atau membuat kebijakan kustom Anda sendiri. Untuk izin minimum yang harus Anda gunakan dalam kebijakan kustom Anda, lihat [the section called “IAM role simpul”](#). Lampirkan kebijakan terkelola Amazon EKS yang diberi nama AmazonEKSClusterPolicy ke peran tersebut. Untuk melampirkan kebijakan IAM ke kepala sekolah [IAM, prinsipal](#) yang melampirkan kebijakan harus diberikan salah satu tindakan IAM berikut (izin): atau. iam:AttachUserPolicy iam:AttachRolePolicy

```

aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy \
  --role-name myAmazonEKSClusterRole

```

Langkah 2: Buat cluster berkemampuan node hybrid

Anda dapat membuat cluster dengan menggunakan:

- [eksctl](#)
- [AWS CloudFormation](#)
- [AWS CLI](#)
- [AWS Management Console](#)

Buat cluster berkemampuan node hybrid - eksctl

Anda perlu menginstal versi terbaru dari alat baris perintah eksctl. Untuk menginstal atau memperbarui eksctl, lihat [Instalasi](#) dalam eksctl dokumentasi.

1. Buat `cluster-config.yaml` untuk menentukan cluster Amazon IPv4 EKS berkemampuan node hybrid. Buat penggantian berikut di `cluster-config.yaml`. Untuk daftar lengkap pengaturan, lihat dokumentasi [eksctl](#).
 - a. Ganti `CLUSTER_NAME` dengan nama untuk cluster Anda. Nama hanya dapat berisi karakter alfanumerik (peka huruf besar/kecil) dan tanda hubung. Itu harus dimulai dengan karakter alfanumerik dan tidak boleh lebih dari 100 karakter. Nama harus unik di dalam AWS Wilayah dan AWS akun tempat Anda membuat klaster.
 - b. Ganti `AWS_REGION` dengan AWS Region tempat Anda ingin membuat cluster Anda.
 - c. Ganti `K8S_VERSION` dengan [versi yang didukung Amazon EKS](#).
 - d. Ganti `CREDS_PROVIDER` dengan `ssm` atau `ira` berdasarkan penyedia kredensi yang Anda konfigurasi dalam langkah-langkahnya. [the section called “Siapkan kredensial”](#)
 - e. Ganti `CA_BUNDLE_CERT` jika penyedia kredensi Anda disetel ke `ira`, yang menggunakan AWS IAM Roles Anywhere sebagai penyedia kredensi. `CA_BUNDLE_CERT` adalah badan sertifikat otoritas sertifikat (CA) dan tergantung pada pilihan CA Anda. Sertifikat harus dalam format Privacy Enhanced Mail (PEM).
 - f. Ganti `GATEWAY_ID` dengan ID gateway pribadi virtual atau gateway transit Anda untuk dilampirkan ke VPC Anda.
 - g. Ganti `REMOTE_NODE_CIDRS` dengan simpul lokal CIDR untuk node hibrid Anda.
 - h. Ganti `REMOTE_POD_CIDRS` dengan pod CIDR lokal untuk beban kerja yang berjalan pada node hibrid atau hapus baris dari konfigurasi Anda jika Anda tidak menjalankan webhook pada node hibrid. Anda harus mengonfigurasi `REMOTE_POD_CIDRS` jika CNI Anda tidak menggunakan Network Address Translation (NAT) atau menyamar untuk alamat IP pod saat lalu lintas pod meninggalkan host lokal Anda. Anda harus mengonfigurasi `REMOTE_POD_CIDRS` jika Anda menjalankan webhook pada node hibrid, lihat [the section called “Konfigurasi webhook”](#) untuk informasi lebih lanjut.
 - i. Blok CIDR node dan pod lokal Anda harus memenuhi persyaratan berikut:
 - i. Berada dalam salah satu rentang IPv4 RFC-1918: `10.0.0.0/8`, atau `172.16.0.0/12` atau `192.168.0.0/16`
 - ii. Tidak tumpang tindih satu sama lain, VPC CIDR untuk klaster Anda, atau CIDR layanan Kubernetes Anda IPv4

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: CLUSTER_NAME
  region: AWS_REGION
  version: "K8S_VERSION"

remoteNetworkConfig:
  iam:
    provider: CREDS_PROVIDER # default SSM, can also be set to IRA
    # caBundleCert: CA_BUNDLE_CERT
  vpcGatewayID: GATEWAY_ID
  remoteNodeNetworks:
    - cidrs: ["REMOTE_NODE_CIDRS"]
  remotePodNetworks:
    - cidrs: ["REMOTE_POD_CIDRS"]
```

2. Jalankan perintah berikut:

```
eksctl create cluster -f cluster-config.yaml
```

Penyediaan kluster memerlukan waktu beberapa menit. Saat cluster sedang dibuat, beberapa baris output muncul. Baris terakhir output mirip dengan baris contoh berikut.

```
[#] EKS cluster "CLUSTER_NAME" in "REGION" region is ready
```

3. Lanjutkan dengan [the section called “Langkah 3: Perbarui kubeconfig”](#).

Buat cluster berkemampuan node hybrid - AWS CloudFormation

CloudFormation Tumpukan membuat peran IAM kluster EKS dan cluster EKS dengan RemoteNodeNetwork dan RemotePodNetwork Anda tentukan. Ubah CloudFormation template Jika Anda perlu menyesuaikan pengaturan untuk kluster EKS Anda yang tidak terekspos di CloudFormation template.

1. Unduh CloudFormation template.

```
curl -OL 'https://raw.githubusercontent.com/aws/eks-hybrid/refs/heads/main/example/hybrid-eks-cfn.yaml'
```

2. Buat `cfn-eks-parameters.json` dan tentukan konfigurasi Anda untuk setiap nilai.

- a. `CLUSTER_NAME`: nama cluster EKS yang akan dibuat
- b. `CLUSTER_ROLE_NAME`: nama peran IAM cluster EKS yang akan dibuat. Default dalam template adalah “EKSClusterPeran”.
- c. `SUBNET1_ID`: ID subnet pertama yang Anda buat dalam langkah-langkah prasyarat
- d. `SUBNET2_ID`: ID subnet kedua yang Anda buat dalam langkah-langkah prasyarat
- e. `SG_ID`: ID grup keamanan yang Anda buat dalam langkah-langkah prasyarat
- f. `REMOTE_NODE_CIDRS`: simpul lokal CIDR untuk node hibrida Anda
- g. `REMOTE_POD_CIDRS`: pod CIDR lokal untuk beban kerja yang berjalan pada node hybrid. Anda harus mengonfigurasi `REMOTE_POD_CIDRS` jika CNi Anda tidak menggunakan Network Address Translation (NAT) atau menyamar untuk alamat IP pod saat lalu lintas pod meninggalkan host lokal Anda. Anda harus mengonfigurasi `REMOTE_POD_CIDRS` jika Anda menjalankan webhook pada node hybrid, lihat [the section called “Konfigurasi webhook”](#) untuk informasi lebih lanjut.
- h. Blok CIDR node dan pod lokal Anda harus memenuhi persyaratan berikut:
 - i. Berada dalam salah satu rentang IPv4 RFC-1918: `10.0.0.0/8`, atau `172.16.0.0/12` atau `192.168.0.0/16`
 - ii. Tidak tumpang tindih satu sama lain, VPC CIDR untuk kluster Anda, atau CIDR layanan Kubernetes Anda. IPv4
- i. `CLUSTER_AUTH`: mode otentikasi cluster untuk cluster Anda. Nilai yang valid adalah `API` dan `API_AND_CONFIG_MAP`. Default dalam template adalah `API_AND_CONFIG_MAP`.
- j. `CLUSTER_ENDPOINT`: konektivitas titik akhir cluster untuk cluster Anda. Nilai yang valid adalah “Publik” dan “Pribadi”. Default dalam template adalah `Private`, yang berarti Anda hanya akan dapat terhubung ke endpoint API Kubernetes dari dalam VPC Anda.
- k. `K8S_VERSION`: versi Kubernetes yang akan digunakan untuk kluster Anda. Lihat [the section called “Versi Kubernetes”](#).

```
{
  "Parameters": {
    "ClusterName": "CLUSTER_NAME",
    "ClusterRoleName": "CLUSTER_ROLE_NAME",
```



```
"SubnetId1": "SUBNET1_ID",
"SubnetId2": "SUBNET2_ID",
"SecurityGroupId": "SG_ID",
"RemoteNodeCIDR": "REMOTE_NODE_CIDRS",
"RemotePodCIDR": "REMOTE_POD_CIDRS",
"ClusterAuthMode": "CLUSTER_AUTH",
"ClusterEndpointConnectivity": "CLUSTER_ENDPOINT",
"K8sVersion": "K8S_VERSION"
}
}
```

3. Menyebarkan CloudFormation tumpukan. Ganti `STACK_NAME` dengan nama Anda untuk CloudFormation tumpukan dan `AWS_REGION` dengan AWS Wilayah Anda di mana cluster akan dibuat.

```
aws cloudformation deploy \
  --stack-name STACK_NAME \
  --region AWS_REGION \
  --template-file hybrid-eks-cfn.yaml \
  --parameter-overrides file://cfn-eks-parameters.json \
  --capabilities CAPABILITY_NAMED_IAM
```

Penyediaan klaster memerlukan waktu beberapa menit. Anda dapat memeriksa status tumpukan Anda dengan perintah berikut. Ganti `STACK_NAME` dengan nama Anda untuk CloudFormation tumpukan dan `AWS_REGION` dengan AWS Wilayah Anda di mana cluster akan dibuat.

```
aws cloudformation describe-stacks \
  --stack-name STACK_NAME \
  --region AWS_REGION \
  --query 'Stacks[].StackStatus'
```

4. Lanjutkan dengan [the section called “Langkah 3: Perbarui kubeconfig”](#).

Buat cluster berkemampuan node hybrid - CLI AWS

1. Jalankan perintah berikut untuk membuat cluster EKS berkemampuan node hybrid. Sebelum menjalankan perintah, ganti yang berikut ini dengan pengaturan Anda. Untuk daftar lengkap pengaturan, lihat [the section called “Membuat klaster”](#) dokumentasi.
 - a. `CLUSTER_NAME`: nama cluster EKS yang akan dibuat
 - b. `AWS_REGION`: AWS Wilayah tempat cluster akan dibuat.

- c. K8S_VERSION: versi Kubernetes yang akan digunakan untuk kluster Anda. Lihat versi yang didukung Amazon EKS.
- d. ROLE_ARN: peran kluster Amazon EKS yang Anda konfigurasi untuk cluster Anda. Lihat peran IAM kluster Amazon EKS untuk informasi selengkapnya.
- e. SUBNET1_ID: ID subnet pertama yang Anda buat dalam langkah-langkah prasyarat
- f. SUBNET2_ID: ID subnet kedua yang Anda buat dalam langkah-langkah prasyarat
- g. SG_ID: ID grup keamanan yang Anda buat dalam langkah-langkah prasyarat
- h. Anda dapat menggunakan API dan API_AND_CONFIG_MAP untuk mode otentikasi akses cluster Anda. Pada perintah di bawah ini, mode otentikasi akses cluster diatur ke API_AND_CONFIG_MAP.
- i. Anda dapat menggunakan endpointPrivateAccess parameter endpointPublicAccess dan untuk mengaktifkan atau menonaktifkan akses publik dan pribadi ke titik akhir server Kubernetes API kluster Anda. Dalam perintah di bawah endpointPublicAccess ini diatur ke false dan endpointPrivateAccess diatur ke true.
- j. REMOTE_NODE_CIDRS: simpul lokal CIDR untuk node hybrid Anda.
- k. REMOTE_POD_CIDRS(optional): CIDR pod lokal untuk beban kerja yang berjalan pada node hybrid.
- l. Blok CIDR node dan pod lokal Anda harus memenuhi persyaratan berikut:
 - i. Berada dalam salah satu rentang IPv4 RFC-1918:10.0.0.0/8,, atau. 172.16.0.0/12 192.168.0.0/16
 - ii. Tidak tumpang tindih satu sama lain, VPC CIDR untuk cluster Amazon EKS Anda, atau CIDR layanan Kubernetes Anda. IPv4

```
aws eks create-cluster \
  --name CLUSTER_NAME \
  --region AWS_REGION \
  --kubernetes-version K8S_VERSION \
  --role-arn ROLE_ARN \
  --resources-vpc-config
subnetIds=SUBNET1_ID,SUBNET2_ID,securityGroupIds=SG_ID,endpointPrivateAccess=true,endpointPublicAccess=false \
  --access-config authenticationMode=API_AND_CONFIG_MAP \
  --remote-network-config '{"remoteNodeNetworks":[{"cidrs":
["REMOTE_NODE_CIDRS"]}], "remotePodNetworks":[{"cidrs":["REMOTE_POD_CIDRS"]}]}'
```

2. Dibutuhkan beberapa menit untuk menyediakan cluster. Anda dapat melakukan kueri status klaster Anda dengan perintah berikut. Ganti CLUSTER_NAME dengan nama cluster yang Anda buat dan AWS_REGION dengan AWS Wilayah tempat cluster dibuat. Jangan lanjutkan ke langkah berikutnya sampai output yang dikembalikan ACTIVE.

```
aws eks describe-cluster \
  --name CLUSTER_NAME \
  --region AWS_REGION \
  --query "cluster.status"
```

3. Lanjutkan dengan [the section called “Langkah 3: Perbarui kubeconfig”](#).

Buat cluster berkemampuan node hybrid - AWS Management Console

1. Buka konsol Amazon EKS di konsol [Amazon EKS](#).
2. Pilih Add cluster dan kemudian pilih Create.
3. Pada halaman Configure cluster, masukkan bidang-bidang berikut:
 - a. Nama — Nama untuk klaster Anda. Nama hanya dapat berisi karakter alfanumerik (peka huruf besar/kecil), tanda hubung, dan garis bawah. Itu harus dimulai dengan karakter alfanumerik dan tidak boleh lebih dari 100 karakter. Nama harus unik di dalam AWS Wilayah dan AWS akun tempat Anda membuat klaster.
 - b. Peran IAM Cluster — Pilih peran IAM klaster Amazon EKS yang Anda buat untuk memungkinkan bidang kontrol Kubernetes mengelola AWS sumber daya atas nama Anda.
 - c. Versi Kubernetes — Versi Kubernetes untuk digunakan untuk klaster Anda. Sebaiknya pilih versi terbaru, kecuali jika Anda memerlukan versi sebelumnya.
 - d. Kebijakan peningkatan - Pilih Extended atau Standard.
 - i. Diperpanjang: Opsi ini mendukung versi Kubernetes selama 26 bulan setelah tanggal rilis. Periode dukungan yang diperpanjang memiliki biaya per jam tambahan yang dimulai setelah periode dukungan standar berakhir. Ketika dukungan diperpanjang berakhir, cluster Anda akan otomatis ditingkatkan ke versi berikutnya.
 - ii. Standar: Opsi ini mendukung versi Kubernetes selama 14 bulan setelah tanggal rilis. Tidak ada biaya tambahan. Ketika dukungan standar berakhir, cluster Anda akan otomatis ditingkatkan ke versi berikutnya.
 - e. Akses cluster - pilih untuk mengizinkan atau melarang akses administrator cluster dan pilih mode otentikasi. Mode otentikasi berikut didukung untuk cluster berkemampuan node hybrid.

- i. EKS API: Cluster akan mencari prinsip IAM yang diautentikasi hanya dari entri akses EKS. APIs
 - ii. EKS API dan ConfigMap: Cluster akan mendapatkan prinsipal IAM yang diautentikasi dari entri akses EKS dan. APIs aws-auth ConfigMap
- f. Enkripsi rahasia — (Opsional) Pilih untuk mengaktifkan enkripsi rahasia Kubernetes menggunakan kunci KMS. Anda juga dapat mengaktifkan ini setelah Anda membuat cluster Anda. Sebelum Anda mengaktifkan kemampuan ini, pastikan Anda terbiasa dengan informasi di dalamnya [the section called “Aktifkan enkripsi rahasia”](#).
- g. ARC Zonal Shift - Jika diaktifkan, EKS akan mendaftarkan cluster Anda dengan ARC zonal shift untuk memungkinkan Anda menggunakan pergeseran zona untuk mengalihkan lalu lintas aplikasi dari AZ.
- h. Tanda — (Opsional) Tambahkan tanda apapun ke klaster Anda. Untuk informasi selengkapnya, lihat [the section called “Pemberian tag pada sumber daya Anda”](#).
- i. Setelah selesai dengan halaman ini, pilih Berikutnya.
4. Pada halaman Tentukan jaringan, pilih nilai untuk kolom berikut:
- a. VPC — Pilih VPC yang sudah ada yang memenuhi dan persyaratan [the section called “Persyaratan VPC dan subnet”](#) [Amazon EKS](#) Hybrid Nodes. Sebelum memilih VPC, sebaiknya Anda memahami semua persyaratan dan pertimbangan dalam persyaratan jaringan View Amazon EKS untuk VPC, subnet, dan node hybrid. Anda tidak dapat mengubah VPC mana yang ingin Anda gunakan setelah pembuatan cluster. Jika tidak VPCs terdaftar, maka Anda harus membuatnya terlebih dahulu. Untuk informasi selengkapnya, lihat [the section called “Buat VPC”](#) dan [persyaratan jaringan Amazon EKS Hybrid Nodes](#).
 - b. Subnet — Secara default, semua subnet yang tersedia di VPC yang ditentukan di bidang sebelumnya telah dipilih sebelumnya. Anda harus memilih setidaknya dua.
 - c. Grup keamanan — (Opsional) Tentukan satu atau beberapa grup keamanan yang ingin Anda kaitkan Amazon EKS ke antarmuka jaringan yang dibuatnya. Setidaknya salah satu grup keamanan yang Anda tentukan harus memiliki aturan masuk untuk node lokal dan pod opsional. CIDRs Lihat [persyaratan jaringan Amazon EKS Hybrid Nodes](#) untuk informasi selengkapnya. Apakah Anda memilih grup keamanan atau tidak, Amazon EKS membuat grup keamanan yang memungkinkan komunikasi antara cluster dan VPC Anda. Amazon EKS mengaitkan grup keamanan ini, dan apa pun yang Anda pilih, ke antarmuka jaringan yang dibuatnya. Untuk informasi selengkapnya tentang grup keamanan klaster yang dibuat Amazon EKS, lihat [the section called “Persyaratan grup keamanan”](#) Anda dapat mengubah aturan di grup keamanan klaster yang dibuat Amazon EKS.

- d. Pilih keluarga alamat IP cluster — Anda harus memilih klaster IPv4 berkemampuan node hibrida.
 - e. (Opsional) Pilih Konfigurasi rentang alamat IP Layanan Kubernetes dan tentukan rentang Layanan. IPv4
 - f. Pilih Konfigurasi jaringan jarak jauh untuk mengaktifkan node hibrid dan tentukan node dan pod lokal Anda CIDRs untuk node hybrid.
 - g. Anda harus mengonfigurasi CIDR pod jarak jauh jika CNI Anda tidak menggunakan Network Address Translation (NAT) atau menyamar untuk alamat IP pod saat lalu lintas pod meninggalkan host lokal Anda. Anda harus mengonfigurasi pod jarak jauh CIDR jika Anda menjalankan webhook pada node hybrid.
 - h. Blok CIDR node dan pod lokal Anda harus memenuhi persyaratan berikut:
 - i. Berada dalam salah satu rentang IPv4 RFC-1918:10.0.0.0/8,, atau. 172.16.0.0/12
192.168.0.0/16
 - ii. Tidak tumpang tindih satu sama lain, VPC CIDR untuk klaster Anda, atau CIDR layanan Kubernetes Anda IPv4
 - i. Untuk akses endpoint Cluster, pilih opsi. Setelah cluster Anda dibuat, Anda dapat mengubah opsi ini. Untuk cluster berkemampuan node hybrid, Anda harus memilih Publik atau Pribadi. Sebelum memilih opsi non-default, pastikan untuk membiasakan diri dengan opsi dan implikasinya. Untuk informasi selengkapnya, lihat [the section called “Akses titik akhir klaster”](#).
 - j. Setelah selesai dengan halaman ini, pilih Berikutnya.
5. (Opsional) Pada halaman Konfigurasi observabilitas, pilih opsi pencatatan bidang Metrik dan Kontrol mana yang akan diaktifkan. Secara default, setiap jenis log dimatikan.
- a. Untuk informasi selengkapnya tentang opsi metrik Prometheus, lihat. [the section called “Metrik Prometheus”](#)
 - b. Untuk informasi selengkapnya tentang opsi pencatatan kontrol EKS, lihat [the section called “Kontrol log pesawat”](#).
 - c. Setelah selesai dengan halaman ini, pilih Berikutnya.
6. Pada halaman Pilih add-on, pilih add-on yang ingin Anda tambahkan ke cluster Anda.
- a. Anda dapat memilih add-on Amazon EKS dan add-on AWS Marketplace sebanyak yang Anda butuhkan. Add-on Amazon EKS yang tidak kompatibel dengan node hybrid ditandai dengan “Tidak kompatibel dengan Hybrid Nodes” dan add-on memiliki aturan anti-afinitas untuk mencegahnya berjalan pada node hybrid. Lihat Mengonfigurasi add-on untuk node hybrid untuk informasi selengkapnya. Jika add-on AWS Marketplace yang ingin Anda instal tidak terdaftar,

- Anda dapat mencari add-on AWS Marketplace yang tersedia dengan memasukkan teks di kotak pencarian. Anda juga dapat mencari berdasarkan kategori, vendor, atau model harga dan kemudian memilih add-on dari hasil pencarian.
- b. Beberapa add-on, seperti CoreDNS dan kube-proxy, diinstal secara default. Jika Anda menonaktifkan salah satu add-on default, ini dapat memengaruhi kemampuan Anda untuk menjalankan aplikasi Kubernetes.
 - c. Setelah selesai dengan halaman ini, pilih **Next**.
7. Pada halaman Konfigurasi pengaturan add-on yang dipilih, pilih versi yang ingin Anda instal.
- a. Anda selalu dapat memperbarui ke versi yang lebih baru setelah pembuatan cluster. Anda dapat memperbarui konfigurasi setiap add-on setelah pembuatan cluster. Untuk informasi selengkapnya tentang mengonfigurasi add-on, lihat [the section called “Perbarui add-on”](#). Untuk versi add-on yang kompatibel dengan node hybrid, lihat [the section called “Konfigurasi add-on”](#).
 - b. Setelah selesai dengan halaman ini, pilih **Berikutnya**.
8. Pada halaman Tinjau dan buat, tinjau informasi yang Anda masukkan atau pilih pada halaman sebelumnya. Jika Anda perlu melakukan perubahan, pilih **Edit**. Saat Anda puas, pilih **Buat**. Bidang Status menunjukkan **CREATING** saat cluster disediakan. Penyediaan kluster memerlukan waktu beberapa menit.
9. Lanjutkan dengan [the section called “Langkah 3: Perbarui kubeconfig”](#).

Langkah 3: Perbarui kubeconfig

Jika Anda membuat cluster Anda menggunakan `eksctl`, maka Anda dapat melewati langkah ini. Ini karena `eksctl` sudah menyelesaikan langkah ini untuk Anda. Aktifkan `kubectl` untuk berkomunikasi dengan cluster Anda dengan menambahkan konteks baru ke file `kubectl` konfigurasi. Untuk informasi selengkapnya tentang cara membuat dan memperbarui file, lihat [the section called “Akses kluster dengan kubectl”](#).

```
aws eks update-kubeconfig --name CLUSTER_NAME --region AWS_REGION
```

Contoh output adalah sebagai berikut.

```
Added new context arn:aws:eks:AWS_REGION:111122223333:cluster/CLUSTER_NAME to /home/username/.kube/config
```

Konfirmasikan komunikasi dengan cluster Anda dengan menjalankan perintah berikut.

```
kubectl get svc
```

Contoh output adalah sebagai berikut.

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	ClusterIP	10.100.0.1	<none>	443/TCP	28h

Langkah 4: Pengaturan cluster

Sebagai langkah selanjutnya, lihat [the section called “Siapkan akses cluster”](#) untuk mengaktifkan akses untuk node hybrid Anda untuk bergabung dengan cluster Anda.

Aktifkan node hibrid pada kluster Amazon EKS yang ada atau ubah konfigurasi

Topik ini memberikan ikhtisar opsi yang tersedia dan menjelaskan apa yang harus dipertimbangkan saat Anda menambahkan, mengubah, atau menghapus konfigurasi node hibrida untuk kluster Amazon EKS.

Untuk mengaktifkan kluster Amazon EKS menggunakan node hibrid, tambahkan rentang CIDR alamat IP dari node lokal Anda dan jaringan pod opsional dalam konfigurasi.

RemoteNetworkConfig EKS menggunakan daftar ini CIDRs untuk mengaktifkan konektivitas antara kluster dan jaringan lokal Anda. Untuk daftar lengkap opsi saat memperbarui konfigurasi cluster Anda, lihat [UpdateClusterConfig](#) di Referensi API Amazon EKS.

Anda dapat melakukan salah satu tindakan berikut ke konfigurasi jaringan EKS Hybrid Nodes dalam sebuah cluster:

- [Tambahkan konfigurasi jaringan jarak jauh untuk mengaktifkan EKS Hybrid Nodes di cluster yang ada.](#)
- [Tambahkan, ubah, atau hapus jaringan node jarak jauh atau jaringan pod jarak jauh di cluster yang ada.](#)
- [Hapus semua rentang CIDR jaringan node jarak jauh untuk menonaktifkan EKS Hybrid Nodes di cluster yang ada.](#)

Prasyarat

- Sebelum mengaktifkan kluster Amazon EKS untuk node hibrida, pastikan lingkungan Anda memenuhi persyaratan yang diuraikan di [the section called “Prasyarat”](#), dan dirinci di [the section](#)

[called “Siapkan jaringan”](#), [the section called “Siapkan sistem operasi”](#), dan [the section called “Siapkan kredensial”](#)

- Cluster Anda harus menggunakan keluarga IPv4 alamat.
- Cluster Anda harus menggunakan salah satu API atau API_AND_CONFIG_MAP untuk mode otentikasi cluster. Proses untuk memodifikasi modus otentikasi cluster dijelaskan di [the section called “Mode autentikasi”](#)
- Kami menyarankan Anda menggunakan akses endpoint publik atau pribadi untuk endpoint server API Amazon EKS Kubernetes, tetapi tidak keduanya. Jika Anda memilih “Publik dan Pribadi”, endpoint server API Amazon EKS Kubernetes akan selalu menyelesaikan ke publik IPs untuk node hybrid yang berjalan di luar VPC Anda, yang dapat mencegah node hybrid Anda bergabung dengan cluster. Proses untuk memodifikasi akses jaringan ke cluster Anda dijelaskan di [the section called “Akses titik akhir kluster”](#).
- Versi terbaru dari AWS Command Line Interface (AWS CLI) diinstal dan dikonfigurasi pada perangkat Anda. Untuk memeriksa versi Anda saat ini, gunakan `aws --version`. Package manager seperti yum, apt-get, atau Homebrew untuk macOS seringkali merupakan beberapa versi di belakang versi terbaru CLI. AWS Untuk menginstal versi terbaru, lihat [Menginstal atau memperbarui ke versi terbaru CLI dan Mengkonfigurasi pengaturan untuk AWS CLI di Panduan Pengguna](#) Antarmuka Baris AWS Perintah.
- [Prinsipal IAM](#) dengan izin untuk menelepon [UpdateClusterConfig](#) kluster Amazon EKS Anda.
- Perbarui add-on ke versi yang kompatibel dengan node hybrid. Untuk versi add-on yang kompatibel dengan node hybrid, lihat [the section called “Konfigurasi add-on”](#).
- Jika Anda menjalankan add-on yang tidak kompatibel dengan node hybrid, pastikan bahwa add-on [DaemonSet](#) atau [Deployment](#) memiliki aturan afinitas berikut untuk mencegah penyebaran ke node hybrid. Tambahkan aturan afinitas berikut jika belum ada.

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: eks.amazonaws.com/compute-type
          operator: NotIn
          values:
            - hybrid
```


Pertimbangan

Objek `remoteNetworkConfig` JSON memiliki perilaku berikut selama pembaruan:

- Setiap bagian konfigurasi yang ada yang tidak Anda tentukan tidak berubah. Jika Anda tidak menentukan salah satu dari `remoteNodeNetworks` atau `remotePodNetworks`, bagian itu akan tetap sama.
- Jika Anda memodifikasi `remotePodNetworks` atau `remoteNodeNetworks` atau daftar CIDRs, Anda harus menentukan daftar lengkap CIDRs yang Anda inginkan dalam konfigurasi akhir Anda. Saat Anda menentukan perubahan ke daftar `remoteNodeNetworks` atau `remotePodNetworks` CIDR, EKS menggantikan daftar asli selama pembaruan.
- Blok CIDR node dan pod lokal Anda harus memenuhi persyaratan berikut:
 1. Berada dalam salah satu rentang IPv4 RFC-1918: 10.0.0.0/8, 172.16.0.0/12, atau 192.168.0.0/16.
 2. Tidak saling tumpang tindih, semua CIDRs VPC untuk cluster Amazon EKS Anda, atau CIDR layanan IPv4 Kubernetes Anda.

Aktifkan node hybrid pada cluster yang ada

Anda dapat mengaktifkan EKS Hybrid Nodes di cluster yang ada dengan menggunakan:

- [AWS CloudFormation](#)
- [AWS CLI](#)
- [AWS Management Console](#)

Aktifkan Node Hibrid EKS di cluster yang ada - AWS CloudFormation

1. Untuk mengaktifkan EKS Hybrid Nodes di cluster Anda, tambahkan `RemoteNodeNetwork` dan (opsional) `RemotePodNetwork` ke CloudFormation template Anda dan perbarui tumpukan. Perhatikan bahwa `RemoteNodeNetwork` adalah daftar dengan maksimum satu `Cidrs` item dan `Cidrs` adalah daftar beberapa rentang IP CIDR.

```
RemoteNetworkConfig:
  RemoteNodeNetworks:
    - Cidrs: [RemoteNodeCIDR]
  RemotePodNetworks:
    - Cidrs: [RemotePodCIDR]
```

2. Lanjutkan ke [the section called “Siapkan akses cluster”](#).

Aktifkan Node Hibrid EKS di cluster yang ada - AWS CLI

1. Jalankan perintah berikut `RemoteNetworkConfig` untuk mengaktifkan EKS Hybrid Nodes untuk kluster EKS Anda. Sebelum menjalankan perintah, ganti yang berikut ini dengan pengaturan Anda. Untuk daftar lengkap pengaturan, lihat [UpdateClusterConfig](#) di Referensi API Amazon EKS.
 - a. `CLUSTER_NAME`: nama cluster EKS untuk diperbarui.
 - b. `AWS_REGION`: AWS Wilayah tempat cluster EKS berjalan.
 - c. `REMOTE_NODE_CIDRS`: simpul lokal CIDR untuk node hybrid Anda.
 - d. `REMOTE_POD_CIDRS`(opsional): CIDR pod lokal untuk beban kerja yang berjalan pada node hybrid.

```
aws eks update-cluster-config \
  --name CLUSTER_NAME \
  --region AWS_REGION \
  --remote-network-config '{"remoteNodeNetworks":[{"cidrs":
["REMOTE_NODE_CIDRS"]}],"remotePodNetworks":[{"cidrs":["REMOTE_POD_CIDRS"]}]}'
```

2. Dibutuhkan beberapa menit untuk memperbarui cluster. Anda dapat melakukan kueri status klaster Anda dengan perintah berikut. Ganti `CLUSTER_NAME` dengan nama cluster yang Anda modifikasi dan `AWS_REGION` dengan AWS Wilayah tempat cluster berjalan. Jangan lanjutkan ke langkah berikutnya sampai output yang dikembalikan `ACTIVE`.

```
aws eks describe-cluster \
  --name CLUSTER_NAME \
  --region AWS_REGION \
  --query "cluster.status"
```

3. Lanjutkan ke [the section called “Siapkan akses cluster”](#).

Aktifkan Node Hibrid EKS di cluster yang ada - AWS Management Console

1. Buka konsol Amazon EKS di konsol [Amazon EKS](#).
2. Pilih nama klaster untuk menampilkan informasi klaster Anda.
3. Pilih tab Jaringan dan pilih Kelola.
4. Di dropdown, pilih Jaringan jarak jauh.

5. Pilih Konfigurasi jaringan jarak jauh untuk mengaktifkan node hibrid dan tentukan node dan pod lokal Anda CIDRs untuk node hibrid.
6. Pilih Simpan perubahan untuk menyelesaikan. Tunggu status kluster kembali ke Aktif.
7. Lanjutkan ke [the section called “Siapkan akses cluster”](#).

Perbarui konfigurasi node hibrid di cluster yang ada

Anda dapat memodifikasi `remoteNetworkConfig` di cluster hibrid yang ada dengan menggunakan salah satu dari berikut ini:

- [AWS CloudFormation](#)
- [AWS CLI](#)
- [AWS Management Console](#)

Perbarui konfigurasi hibrid di cluster yang ada - AWS CloudFormation

1. Perbarui CloudFormation template Anda dengan nilai CIDR jaringan baru.

```
RemoteNetworkConfig:
  RemoteNodeNetworks:
    - Cidrs: [NEW_REMOTE_NODE_CIDRS]
  RemotePodNetworks:
    - Cidrs: [NEW_REMOTE_POD_CIDRS]
```

Note

Saat memperbarui `RemoteNodeNetworks` atau daftar `RemotePodNetworks` CIDR, sertakan semua CIDRs (baru dan yang sudah ada). EKS menggantikan seluruh daftar selama pembaruan. Menghilangkan bidang ini dari permintaan pembaruan mempertahankan konfigurasi yang ada.

2. Perbarui CloudFormation tumpukan Anda dengan template yang dimodifikasi dan tunggu pembaruan tumpukan selesai.

Perbarui konfigurasi hybrid di cluster yang ada - AWS CLI

1. Untuk memodifikasi jaringan jarak jauh CIDRs, jalankan perintah berikut. Ganti nilai dengan pengaturan Anda:

```
aws eks update-cluster-config
--name CLUSTER_NAME
--region AWS_REGION
--remote-network-config '{"remoteNodeNetworks":[{"cidrs":
["NEW_REMOTE_NODE_CIDRS"]}], "remotePodNetworks":[{"cidrs":
["NEW_REMOTE_POD_CIDRS"]}]}'
```

Note

Saat memperbarui `remoteNodeNetworks` atau daftar `remotePodNetworks` CIDR, sertakan semua CIDRs (baru dan yang sudah ada). EKS menggantikan seluruh daftar selama pembaruan. Menghilangkan bidang ini dari permintaan pembaruan mempertahankan konfigurasi yang ada.

2. Tunggu status cluster kembali ke ACTIVE sebelum melanjutkan.

Perbarui konfigurasi hybrid di cluster yang ada - AWS Management Console

1. Buka konsol Amazon EKS di konsol [Amazon EKS](#).
2. Pilih nama klaster untuk menampilkan informasi klaster Anda.
3. Pilih tab Jaringan dan pilih Kelola.
4. Di dropdown, pilih Jaringan jarak jauh.
5. Perbarui bagian CIDRs bawah Remote node networks dan Remote pod networks - Optional sesuai kebutuhan.
6. Pilih Simpan perubahan dan tunggu status klaster kembali ke Aktif.

Nonaktifkan node Hybrid di cluster yang ada

Anda dapat menonaktifkan EKS Hybrid Nodes di cluster yang ada dengan menggunakan:

- [AWS CloudFormation](#)
- [AWS CLI](#)

- [AWS Management Console](#)

Nonaktifkan EKS Hybrid Nodes di cluster yang ada - AWS CloudFormation

1. Untuk menonaktifkan EKS Hybrid Nodes di cluster Anda, atur `RemoteNodeNetworks` dan `RemotePodNetworks` kosongkan array di CloudFormation template Anda dan perbarui tumpukan.

```
RemoteNetworkConfig:
  RemoteNodeNetworks: []
  RemotePodNetworks: []
```

Nonaktifkan EKS Hybrid Nodes di cluster yang ada - AWS CLI

1. Jalankan perintah berikut untuk menghapus `RemoteNetworkConfig` dari kluster EKS Anda. Sebelum menjalankan perintah, ganti yang berikut ini dengan pengaturan Anda. Untuk daftar lengkap pengaturan, lihat [UpdateClusterConfig](#) di Referensi API Amazon EKS.

- a. `CLUSTER_NAME`: nama cluster EKS untuk diperbarui.
- b. `AWS_REGION`: AWS Wilayah tempat cluster EKS berjalan.

```
aws eks update-cluster-config \
  --name CLUSTER_NAME \
  --region AWS_REGION \
  --remote-network-config '{"remoteNodeNetworks":[],"remotePodNetworks":[]}'
```

2. Dibutuhkan beberapa menit untuk memperbarui cluster. Anda dapat melakukan kueri status kluster Anda dengan perintah berikut. Ganti `CLUSTER_NAME` dengan nama cluster yang Anda modifikasi dan `AWS_REGION` dengan AWS Wilayah tempat cluster berjalan. Jangan lanjutkan ke langkah berikutnya sampai output yang dikembalikan `ACTIVE`.

```
aws eks describe-cluster \
  --name CLUSTER_NAME \
  --region AWS_REGION \
  --query "cluster.status"
```

Nonaktifkan EKS Hybrid Nodes di cluster yang ada - AWS Management Console

1. Buka konsol Amazon EKS di konsol [Amazon EKS](#).
2. Pilih nama klaster untuk menampilkan informasi klaster Anda.
3. Pilih tab Jaringan dan pilih Kelola.
4. Di dropdown, pilih Jaringan jarak jauh.
5. Pilih Konfigurasi jaringan jarak jauh untuk mengaktifkan node hibrida dan menghapus semua bagian CIDRs bawah Remote node networks dan Remote pod networks - Optional.
6. Pilih Simpan perubahan untuk menyelesaikan. Tunggu status klaster kembali ke Aktif.

Mempersiapkan akses cluster untuk node hybrid

Sebelum menghubungkan node hybrid ke cluster Amazon EKS Anda, Anda harus mengaktifkan Peran IAM Hybrid Nodes Anda dengan izin Kubernetes untuk bergabung dengan cluster. Lihat [the section called “Siapkan kredensial”](#) untuk informasi tentang cara membuat peran IAM Hybrid Nodes. Amazon EKS mendukung dua cara untuk mengaitkan prinsipal IAM dengan Kubernetes Role-Based Access Control (RBAC), entri akses Amazon EKS, dan entri akses. aws-auth ConfigMap Untuk informasi selengkapnya tentang manajemen akses Amazon EKS, lihat [the section called “Akses API Kubernetes”](#).

Gunakan prosedur di bawah ini untuk mengaitkan peran IAM Hybrid Nodes Anda dengan izin Kubernetes. Untuk menggunakan entri akses Amazon EKS, klaster Anda harus dibuat dengan mode API_AND_CONFIG_MAP autentikasi API atau. Untuk menggunakan aws-auth ConfigMap, cluster Anda harus dibuat dengan mode API_AND_CONFIG_MAP otentikasi. Mode otentikasi CONFIG_MAP - only tidak didukung untuk cluster Amazon EKS yang mendukung node hybrid.

Menggunakan entri akses Amazon EKS untuk peran IAM Hybrid Nodes

Ada jenis entri akses Amazon EKS untuk node hybrid bernama HYBRID_LINUX yang dapat digunakan dengan peran IAM. Dengan jenis entri akses ini, nama pengguna secara otomatis diatur ke system:node: {{SessionName}}. Untuk informasi selengkapnya tentang membuat entri akses, lihat [the section called “Buat entri akses”](#).

AWS CLI

1. Anda harus menginstal dan mengonfigurasi AWS CLI versi terbaru di perangkat Anda. Untuk memeriksa versi Anda saat ini, gunakan `aws --version`. Package manager seperti yum, apt-get,

atau Homebrew untuk macOS seringkali merupakan beberapa versi di belakang versi terbaru CLI. AWS Untuk menginstal versi terbaru, lihat [Menginstal dan Konfigurasi cepat dengan aws configure](#) di Panduan Pengguna Antarmuka Baris AWS Perintah.

2. Buat entri akses Anda dengan perintah berikut. Ganti CLUSTER_NAME dengan nama cluster Anda dan HYBRID_NODES_ROLE_ARN dengan ARN dari peran yang Anda buat dalam langkah-langkah untuk [the section called “Siapkan kredensial”](#)

```
aws eks create-access-entry --cluster-name CLUSTER_NAME \  
  --principal-arn HYBRID_NODES_ROLE_ARN \  
  --type HYBRID_LINUX
```

AWS Management Console

1. Buka konsol Amazon EKS di konsol [Amazon EKS](#).
2. Pilih nama cluster berkemampuan node hybrid Anda.
3. Pilih tab Access.
4. Pilih Buat entri akses.
5. Untuk prinsipal IAM, pilih peran IAM Hybrid Nodes yang Anda buat di langkah-langkahnya. [the section called “Siapkan kredensial”](#)
6. Untuk Type, pilih Hybrid Linux.
7. (Opsional) Untuk Tag, tetapkan label ke entri akses. Misalnya, untuk membuatnya lebih mudah untuk menemukan semua sumber daya dengan tag yang sama.
8. Pilih Lewati untuk meninjau dan membuat. Anda tidak dapat menambahkan kebijakan ke entri akses Hybrid Linux atau mengubah cakupan aksesnya.
9. Tinjau konfigurasi untuk entri akses Anda. Jika ada yang terlihat salah, pilih Sebelumnya untuk kembali melalui langkah-langkah dan memperbaiki kesalahan. Jika konfigurasi sudah benar, pilih Buat.

Menggunakan aws-auth ConfigMap untuk peran IAM Hybrid Nodes

Dalam langkah-langkah berikut, Anda akan membuat atau memperbarui aws-auth ConfigMap dengan ARN dari Peran IAM Hybrid Nodes yang Anda buat dalam langkah-langkah untuk [the section called “Siapkan kredensial”](#)

1. Periksa untuk melihat apakah Anda memiliki yang ada `aws-auth ConfigMap` untuk cluster Anda. Perhatikan bahwa jika Anda menggunakan `kubeconfig` file tertentu, gunakan `--kubeconfig` bendera.

```
kubectl describe configmap -n kube-system aws-auth
```

2. Jika Anda ditampilkan `aws-auth ConfigMap`, maka perbarui sesuai kebutuhan.
 - a. Buka `ConfigMap` untuk mengedit.

```
kubectl edit -n kube-system configmap/aws-auth
```

- b. Tambahkan `mapRoles` entri baru sesuai kebutuhan. Ganti `HYBRID_NODES_ROLE_ARN` dengan ARN peran IAM Hybrid Nodes Anda. Catatan, `{{SessionName}}` adalah format template yang benar untuk disimpan di `ConfigMap`. Jangan menggantinya dengan nilai lain.

```
data:
  mapRoles: |
    - groups:
      - system:bootstrappers
      - system:nodes
    rolearn: HYBRID_NODES_ROLE_ARN
    username: system:node:{{SessionName}}
```

- c. Simpan file dan keluar dari editor teks Anda.
3. Jika tidak ada yang ada `aws-auth ConfigMap` untuk cluster Anda, buat dengan perintah berikut. Ganti `HYBRID_NODES_ROLE_ARN` dengan ARN peran IAM Hybrid Nodes Anda. Perhatikan bahwa itu `{{SessionName}}` adalah format template yang benar untuk disimpan di `ConfigMap`. Jangan menggantinya dengan nilai lain.

```
kubectl apply -f=/dev/stdin <<-EOF
apiVersion: v1
kind: ConfigMap
metadata:
  name: aws-auth
  namespace: kube-system
data:
  mapRoles: |
    - groups:
      - system:bootstrappers
      - system:nodes
```



```
rolearn: HYBRID_NODES_ROLE_ARN
username: system:node:{{SessionName}}
EOF
```

Jalankan beban kerja lokal pada node hibrid

Di kluster EKS dengan node hibrid yang diaktifkan, Anda dapat menjalankan aplikasi lokal dan edge di infrastruktur Anda sendiri dengan kluster, fitur, dan alat Amazon EKS yang sama yang Anda gunakan di AWS Cloud.

Bagian berikut berisi step-by-step instruksi untuk menggunakan node hibrid.

Topik

- [Connect node hybrid](#)
- [Hubungkan node hybrid dengan Bottlerocket](#)
- [Tingkatkan node hybrid untuk klaster Anda](#)
- [Pembaruan keamanan patch untuk node hybrid](#)
- [Hapus node hybrid](#)

Connect node hybrid

Note

Langkah-langkah berikut berlaku untuk node hibrid yang menjalankan sistem operasi yang kompatibel kecuali Bottlerocket. Untuk langkah-langkah untuk menghubungkan node hibrid yang menjalankan Bottlerocket, lihat. [the section called “Hubungkan node hybrid dengan Bottlerocket”](#)

Topik ini menjelaskan cara menghubungkan node hibrid ke cluster Amazon EKS. Setelah node hibrid Anda bergabung dengan cluster, mereka akan muncul dengan status Not Ready di konsol Amazon EKS dan di tooling yang kompatibel dengan Kubernetes seperti kubectl. Setelah menyelesaikan langkah-langkah di halaman ini, lanjutkan [the section called “Konfigurasi CNI”](#) untuk membuat node hibrida Anda siap untuk menjalankan aplikasi.

Prasyarat

Sebelum menghubungkan node hybrid ke cluster Amazon EKS Anda, pastikan Anda telah menyelesaikan langkah-langkah prasyarat.

- Anda memiliki konektivitas jaringan dari lingkungan lokal ke AWS Wilayah yang menghosting kluster Amazon EKS Anda. Untuk informasi selengkapnya, lihat [the section called “Siapkan jaringan”](#).
- Anda memiliki sistem operasi yang kompatibel untuk node hybrid yang diinstal pada host lokal Anda. Untuk informasi selengkapnya, lihat [the section called “Siapkan sistem operasi”](#).
- Anda telah membuat peran IAM Hybrid Nodes dan menyiapkan penyedia kredensi lokal (aktivasi hybrid AWS Systems Manager atau AWS IAM Roles Anywhere). Untuk informasi selengkapnya, lihat [the section called “Siapkan kredensial”](#).
- Anda telah membuat cluster Amazon EKS yang mendukung node hybrid Anda. Untuk informasi selengkapnya, lihat [the section called “Buat cluster”](#).
- Anda telah mengaitkan peran IAM Hybrid Nodes Anda dengan izin Kubernetes Role-Based Access Control (RBAC). Untuk informasi selengkapnya, lihat [the section called “Siapkan akses cluster”](#).

Langkah 1: Instal node hybrid CLI (**nodeadm**) di setiap host lokal

Jika Anda menyertakan Amazon EKS Hybrid Nodes CLI (nodeadm) dalam gambar sistem operasi pra-bangun Anda, Anda dapat melewati langkah ini. Untuk informasi lebih lanjut tentang versi node hibridanodeadm, lihat[the section called “Node hibrida nodeadm”](#).

Versi node hibrida di-host di Amazon S3 yang digawangi oleh Amazon. nodeadm CloudFront Untuk menginstal nodeadm di setiap host lokal, Anda dapat menjalankan perintah berikut dari host lokal Anda.

Untuk host x86_64:

```
curl -OL 'https://hybrid-assets.eks.amazonaws.com/releases/latest/bin/linux/amd64/nodeadm'
```

Untuk host ARM

```
curl -OL 'https://hybrid-assets.eks.amazonaws.com/releases/latest/bin/linux/arm64/nodeadm'
```

Tambahkan izin file yang dapat dieksekusi ke biner yang diunduh di setiap host.

```
chmod +x nodeadm
```

Langkah 2: Instal dependensi node hybrid dengan **nodeadm**

Jika Anda menginstal dependensi node hybrid dalam gambar sistem operasi pra-bangun, Anda dapat melewati langkah ini. `nodeadm install` Perintah ini dapat digunakan untuk menginstal semua dependensi yang diperlukan untuk node hybrid. Dependensi node hybrid mencakup komponen `containerd`, `kubelet`, `kubectl`, dan SSM atau IAM Roles Anywhere. AWS Lihat [the section called “Node hibrida nodeadm”](#) untuk informasi selengkapnya tentang komponen dan lokasi file yang diinstal oleh `nodeadm install`. Lihat node hibrid [the section called “Siapkan jaringan”](#) untuk informasi selengkapnya tentang domain yang harus diizinkan di firewall lokal untuk proses tersebut `nodeadm install`.

Jalankan perintah di bawah ini untuk menginstal dependensi node hybrid pada host lokal Anda. Perintah di bawah ini harus dijalankan dengan pengguna yang memiliki akses `sudo`/`root` pada host Anda.

Important

Node hybrid CLI (`nodeadm`) harus dijalankan dengan pengguna yang memiliki akses `sudo`/`root` pada host Anda.

- Ganti `K8S_VERSION` dengan versi minor Kubernetes dari cluster Amazon EKS Anda, misalnya, 1.31. Lihat [the section called “Versi Kubernetes”](#) daftar versi Kubernetes yang didukung.
- Ganti `CREDS_PROVIDER` dengan penyedia kredensi lokal yang Anda gunakan. Nilai yang valid adalah `ssm` untuk AWS SSM dan `iam-ra` untuk Peran AWS IAM Di Mana Saja.

```
nodeadm install K8S_VERSION --credential-provider CREDS_PROVIDER
```

Langkah 3: Hubungkan node hybrid ke cluster Anda

Sebelum menghubungkan node hibrid ke klaster, pastikan Anda telah mengizinkan akses yang diperlukan di firewall lokal dan di grup keamanan untuk klaster Anda untuk bidang kontrol Amazon

EKS ke/dari komunikasi node hibrid. Sebagian besar masalah pada langkah ini terkait dengan konfigurasi firewall, konfigurasi grup keamanan, atau konfigurasi peran IAM Hybrid Nodes.

⚠ Important

Node hybrid CLI (nodeadm) harus dijalankan dengan pengguna yang memiliki akses sudo/root pada host Anda.

1. Buat `nodeConfig.yaml` file di setiap host dengan nilai untuk penerapan Anda. Untuk deskripsi lengkap tentang pengaturan konfigurasi yang tersedia, lihat [the section called “Node hibrida nodeadm”](#). Jika peran IAM Hybrid Nodes Anda tidak memiliki izin untuk `eks:DescribeCluster` tindakan tersebut, Anda harus meneruskan endpoint Kubernetes API, bundel CA cluster, dan CIDR layanan IPv4 Kubernetes di bagian cluster Anda. `nodeConfig.yaml`
 - a. Gunakan `nodeConfig.yaml` contoh di bawah ini jika Anda menggunakan aktivasi hibrida AWS SSM untuk penyedia kredensi lokal Anda.
 - i. Ganti `CLUSTER_NAME` dengan nama klaster Anda.
 - ii. Ganti `AWS_REGION` dengan AWS Region hosting cluster Anda. Misalnya, `us-west-2`.
 - iii. Ganti `ACTIVATION_CODE` dengan kode aktivasi yang Anda terima saat membuat aktivasi hibrida AWS SSM Anda. Untuk informasi selengkapnya, lihat [the section called “Siapkan kredensial”](#).
 - iv. Ganti `ACTIVATION_ID` dengan ID aktivasi yang Anda terima saat membuat aktivasi hibrida AWS SSM Anda. Anda dapat mengambil informasi ini dari konsol AWS Systems Manager atau dari perintah AWS `aws ssm describe-activations` CLI.

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: CLUSTER_NAME
    region: AWS_REGION
  hybrid:
    ssm:
      activationCode: ACTIVATION_CODE
      activationId: ACTIVATION_ID
```

- b. Gunakan `nodeConfig.yaml` contoh di bawah ini jika Anda menggunakan AWS IAM Roles Anywhere untuk penyedia kredensi lokal Anda.

- i. Ganti CLUSTER_NAME dengan nama klaster Anda.
- ii. Ganti AWS_REGION dengan AWS Region hosting cluster Anda. Misalnya, us-west-2.
- iii. Ganti NODE_NAME dengan nama node Anda. Nama node harus cocok dengan CN sertifikat pada host jika Anda mengonfigurasi kebijakan kepercayaan peran IAM Hybrid Nodes Anda dengan kondisi "sts:RoleSessionName": "\${aws:PrincipalTag/x509Subject/CN}" sumber daya. Yang nodeName Anda gunakan tidak boleh lebih dari 64 karakter.
- iv. Ganti TRUST_ANCHOR_ARN dengan ARN dari jangkar kepercayaan yang Anda konfigurasi dalam langkah-langkah untuk Siapkan kredensial untuk node hibrida.
- v. Ganti PROFILE_ARN dengan ARN dari jangkar kepercayaan yang Anda konfigurasi dalam langkah-langkahnya. [the section called "Siapkan kredensial"](#)
- vi. Ganti ROLE_ARN dengan ARN peran IAM Hybrid Nodes Anda.
- vii. Ganti CERTIFICATE_PATH dengan path di disk ke sertifikat node Anda. Jika Anda tidak menentukannya, defaultnya adalah/etc/iam/pki/server.pem.
- viii. Ganti KEY_PATH dengan jalur di disk ke kunci pribadi sertifikat Anda. Jika Anda tidak menentukannya, defaultnya adalah/etc/iam/pki/server.key.

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: CLUSTER_NAME
    region: AWS_REGION
  hybrid:
    iamRolesAnywhere:
      nodeName: NODE_NAME
      trustAnchorArn: TRUST_ANCHOR_ARN
      profileArn: PROFILE_ARN
      roleArn: ROLE_ARN
      certificatePath: CERTIFICATE_PATH
      privateKeyPath: KEY_PATH
```

2. Jalankan `nodeadm init` perintah dengan Anda `nodeConfig.yaml` untuk menghubungkan node hibrida Anda ke cluster Amazon EKS Anda.

```
nodeadm init -c file://nodeConfig.yaml
```

Jika perintah di atas berhasil diselesaikan, node hybrid Anda telah bergabung dengan cluster Amazon EKS Anda. Anda dapat memverifikasi ini di konsol Amazon EKS dengan menavigasi ke tab Compute untuk klaster Anda ([pastikan kepala sekolah IAM memiliki izin untuk](#) melihat) atau dengan `kubectl get nodes`

 Important

Node Anda akan memiliki status `Not Ready`, yang diharapkan dan karena kurangnya CNI yang berjalan pada node hybrid Anda. Jika node Anda tidak bergabung dengan cluster, lihat [the section called “Pemecahan Masalah”](#).

Langkah 4: Konfigurasi CNI untuk node hybrid

Untuk membuat node hybrid Anda siap menjalankan aplikasi, lanjutkan dengan langkah-langkahnya [the section called “Konfigurasi CNI”](#).

Hubungkan node hybrid dengan Bottlerocket

Topik ini menjelaskan cara menghubungkan node hybrid yang menjalankan Bottlerocket ke cluster Amazon EKS. [Bottlerocket](#) adalah distribusi Linux open source yang disponsori dan didukung oleh AWS Bottlerocket dibuat khusus untuk menampung beban kerja kontainer. Dengan Bottlerocket, Anda dapat meningkatkan ketersediaan penerapan kontainer dan mengurangi biaya operasional dengan mengotomatiskan pembaruan infrastruktur kontainer Anda. Bottlerocket hanya mencakup perangkat lunak penting untuk menjalankan kontainer, yang meningkatkan penggunaan sumber daya, mengurangi ancaman keamanan, dan menurunkan overhead manajemen.

Hanya VMware varian Bottlerocket versi v1.37.0 ke atas yang didukung dengan EKS Hybrid Nodes. VMware varian dari Bottlerocket tersedia untuk Kubernetes versi v1.28 dan di atasnya. Gambar OS untuk varian ini termasuk kubelet, containerd, aws-iam-authenticator dan prasyarat perangkat lunak lainnya untuk EKS Hybrid Nodes. Anda dapat mengonfigurasi komponen ini menggunakan file [pengaturan](#) Bottlerocket yang menyertakan data pengguna yang dikodekan base64 untuk bootstrap Bottlerocket dan wadah admin. Mengkonfigurasi pengaturan ini memungkinkan Bottlerocket untuk menggunakan penyedia kredensi node hybrid Anda untuk mengautentikasi node hybrid ke cluster Anda. Setelah node hibrid Anda bergabung dengan cluster, mereka akan muncul dengan status `Not Ready` di konsol Amazon EKS dan di perkakas yang kompatibel dengan Kubernetes seperti `kubectl`. Setelah menyelesaikan langkah-langkah di halaman ini, lanjutkan [the section called “Konfigurasi CNI”](#) untuk membuat node hibrida Anda siap untuk menjalankan aplikasi.

Prasyarat

Sebelum menghubungkan node hybrid ke cluster Amazon EKS Anda, pastikan Anda telah menyelesaikan langkah-langkah prasyarat.

- Anda memiliki konektivitas jaringan dari lingkungan lokal ke AWS Wilayah yang menghosting kluster Amazon EKS Anda. Untuk informasi selengkapnya, lihat [the section called “Siapkan jaringan”](#).
- Anda telah membuat peran IAM Hybrid Nodes dan menyiapkan penyedia kredensi lokal (aktivasi hybrid AWS Systems Manager atau AWS IAM Roles Anywhere). Untuk informasi selengkapnya, lihat [the section called “Siapkan kredensial”](#).
- Anda telah membuat cluster Amazon EKS yang mendukung node hybrid Anda. Untuk informasi selengkapnya, lihat [the section called “Buat cluster”](#).
- Anda telah mengaitkan peran IAM Hybrid Nodes Anda dengan izin Kubernetes Role-Based Access Control (RBAC). Untuk informasi selengkapnya, lihat [the section called “Siapkan akses cluster”](#).

Langkah 1: Buat file TOMM pengaturan Bottlerocket

Untuk mengkonfigurasi Bottlerocket untuk node hybrid, Anda perlu membuat `settings.toml` file dengan konfigurasi yang diperlukan. Isi file TOMM akan berbeda berdasarkan penyedia kredensi yang Anda gunakan (SSM atau IAM Roles Anywhere). File ini akan diteruskan sebagai data pengguna saat menyediakan instance Bottlerocket.

SSM

Jika Anda menggunakan AWS Systems Manager sebagai penyedia kredensi Anda, buat `settings.toml` file dengan konten berikut:

```
[settings.kubernetes]
cluster-name = "<cluster-name>"
api-server = "<api-server-endpoint>"
cluster-certificate = "<cluster-certificate-authority>"
hostname-override = "<hostname>"
provider-id = "eks-hybrid:///<region>/<cluster-name>/<hostname>"
authentication-mode = "aws"

[settings.network]
hostname = "<hostname>"

[settings.aws]
```

```

region = "<region>"

[settings.kubernetes.node-labels]
"eks.amazonaws.com/compute-type" = "hybrid"
"eks.amazonaws.com/hybrid-credential-provider" = "ssm"

[settings.host-containers.admin]
enabled = true
user-data = "<base64-encoded-admin-container-userdata>"

[settings.bootstrap-containers.eks-hybrid-setup]
mode = "always"
user-data = "<base64-encoded-bootstrap-container-userdata>"

[settings.host-containers.control]
enabled = true

```

Ganti placeholder dengan nilai-nilai berikut:

- **<cluster-name>**: Nama cluster Amazon EKS Anda.
- **<api-server-endpoint>**: Titik akhir server API cluster Anda.
- **<cluster-certificate-authority>**: Bundel CA yang dikodekan base64 dari cluster Anda.
- **<region>**: AWS Wilayah yang menghosting cluster Anda, misalnya "us-east-1".
- **<hostname>**: Nama host dari instance Bottlerocket, yang juga akan dikonfigurasi sebagai nama node. Ini bisa berupa nilai unik pilihan Anda, tetapi harus mengikuti konvensi penamaan [Objek Kubernetes](#). Selain itu, nama host yang Anda gunakan tidak boleh lebih dari 64 karakter. CATATAN: Saat menggunakan penyedia SSM, nama host dan nama node ini akan diganti dengan ID instance terkelola (misalnya, `mi-* ID`) setelah instance terdaftar dengan SSM.
- **<base64-encoded-admin-container-userdata>**: Konten base64 yang dikodekan dari konfigurasi kontainer admin Bottlerocket. Mengaktifkan wadah admin memungkinkan Anda terhubung ke instance Bottlerocket Anda dengan SSH untuk eksplorasi dan debugging sistem. Meskipun ini bukan pengaturan yang diperlukan, kami sarankan untuk mengaktifkannya untuk kemudahan pemecahan masalah. Lihat [dokumentasi kontainer admin Bottlerocket untuk informasi lebih lanjut tentang otentikasi dengan wadah admin](#). Kontainer admin mengambil pengguna SSH dan input kunci dalam format JSON, misalnya,

```

{
  "user": "<ssh-user>",

```



```
"ssh": {
  "authorized-keys": [
    "<ssh-authorized-key>"
  ]
}
```

- `<base64-encoded-bootstrap-container-userdata>`: Konten base64 yang dikodekan dari konfigurasi wadah bootstrap Bottlerocket. Lihat [dokumentasi wadah bootstrap Bottlerocket](#) untuk informasi lebih lanjut tentang konfigurasinya. Container bootstrap bertanggung jawab untuk mendaftarkan instance sebagai Instans Terkelola AWS SSM dan menggabungkannya sebagai node Kubernetes di Cluster Amazon EKS Anda. Data pengguna yang diteruskan ke wadah bootstrap berbentuk pemanggilan perintah yang menerima sebagai masukan kode aktivasi hibrida SSM dan ID yang sebelumnya Anda buat:

```
eks-hybrid-ssm-setup --activation-id=<activation-id> --activation-code=<activation-code> --region=<region>
```

IAM Roles Anywhere

Jika Anda menggunakan AWS IAM Roles Anywhere sebagai penyedia kredensi Anda, buat `settings.toml` file dengan konten berikut:

```
[settings.kubernetes]
cluster-name = "<cluster-name>"
api-server = "<api-server-endpoint>"
cluster-certificate = "<cluster-certificate-authority>"
hostname-override = "<hostname>"
provider-id = "eks-hybrid:///<region>/<cluster-name>/<hostname>"
authentication-mode = "aws"

[settings.network]
hostname = "<hostname>"

[settings.aws]
region = "<region>"
config = "<base64-encoded-aws-config-file>"

[settings.kubernetes.node-labels]
"eks.amazonaws.com/compute-type" = "hybrid"
```

```
"eks.amazonaws.com/hybrid-credential-provider" = "iam-ra"

[settings.host-containers.admin]
enabled = true
user-data = "<base64-encoded-admin-container-userdata>"

[settings.bootstrap-containers.eks-hybrid-setup]
mode = "always"
user-data = "<base64-encoded-bootstrap-container-userdata>"
```

Ganti placeholder dengan nilai-nilai berikut:

- `<cluster-name>`: Nama cluster Amazon EKS Anda.
- `<api-server-endpoint>`: Titik akhir server API cluster Anda.
- `<cluster-certificate-authority>`: Bundel CA yang dikodekan base64 dari cluster Anda.
- `<region>`: AWS Wilayah yang menampung klaster Anda, misalnya, "us-east-1"
- `<hostname>`: Nama host dari instance Bottlerocket, yang juga akan dikonfigurasi sebagai nama node. Ini bisa berupa nilai unik pilihan Anda, tetapi harus mengikuti konvensi penamaan [Objek Kubernetes](#). Selain itu, nama host yang Anda gunakan tidak boleh lebih dari 64 karakter. CATATAN: Saat menggunakan penyedia IAM-RA, nama node harus cocok dengan CN sertifikat pada host jika Anda mengonfigurasi kebijakan kepercayaan peran IAM Hybrid Nodes Anda dengan kondisi sumber daya. "sts:RoleSessionName": "\${aws:PrincipalTag/x509Subject/CN}"
- `<base64-encoded-aws-config-file>`: Konten berkas konfigurasi Anda yang disandikan base64. AWS Isi file harus sebagai berikut:

```
[default]
credential_process = aws_signing_helper credential-process --certificate /root/.aws/
node.crt --private-key /root/.aws/node.key --profile-arn <profile-arn> --role-arn
<role-arn> --trust-anchor-arn <trust-anchor-arn> --role-session-name <role-session-
name>
```

- `<base64-encoded-admin-container-userdata>`: Konten base64 yang dikodekan dari konfigurasi kontainer admin Bottlerocket. Mengaktifkan wadah admin memungkinkan Anda terhubung ke instance Bottlerocket Anda dengan SSH untuk eksplorasi dan debugging sistem. Meskipun ini bukan pengaturan yang diperlukan, kami sarankan untuk mengaktifkannya untuk kemudahan pemecahan masalah. Lihat [dokumentasi kontainer admin Bottlerocket untuk informasi](#)

[lebih lanjut tentang otentikasi dengan wadah admin](#). Kontainer admin mengambil pengguna SSH dan input kunci dalam format JSON, misalnya,

```
{
  "user": "<ssh-user>",
  "ssh": {
    "authorized-keys": [
      "<ssh-authorized-key>"
    ]
  }
}
```

- `<base64-encoded-bootstrap-container-userdata>`: Konten base64 yang dikodekan dari konfigurasi wadah bootstrap Bottlerocket. Lihat [dokumentasi wadah bootstrap Bottlerocket](#) untuk informasi lebih lanjut tentang konfigurasinya. Wadah bootstrap bertanggung jawab untuk membuat sertifikat host IAM Roles Anywhere dan file kunci pribadi sertifikat pada instance. Ini kemudian akan digunakan oleh `aws_signing_helper` untuk mendapatkan kredensi sementara untuk mengautentikasi dengan kluster Amazon EKS Anda. Data pengguna yang diteruskan ke dalam wadah bootstrap berbentuk pemanggilan perintah yang menerima sebagai masukan isi sertifikat dan kunci pribadi yang Anda buat sebelumnya:

```
eks-hybrid-iam-ra-setup --certificate=<certificate> --key=<private-key>
```

Langkah 2: Menyediakan Bottlerocket vSphere VM dengan data pengguna

Setelah Anda membuat file TOMM, berikan sebagai data pengguna selama pembuatan VSphere VM. Perlu diingat bahwa data pengguna harus dikonfigurasi sebelum VM dinyalakan untuk pertama kalinya. Dengan demikian, Anda harus menyediakannya saat membuat instance, atau jika Anda ingin membuat VM sebelumnya, VM harus dalam status PoweredOff sampai Anda mengonfigurasi data pengguna untuk itu. Misalnya, jika menggunakan `govc` CLI:

Membuat VM untuk pertama kalinya

```
govc vm.create \
  -on=true \
  -c=2 \
  -m=4096 \
  -net.adapter=<network-adapter> \
```

```
-net=<network-name> \  
-e guestinfo.userdata.encoding="base64" \  
-e guestinfo.userdata="$(base64 -w0 settings.toml)" \  
-template=<template-name> \  
<vm-name>
```

Memperbarui data pengguna untuk VM yang ada

```
govc vm.create \  
  -on=false \  
  -c=2 \  
  -m=4096 \  
  -net.adapter=<network-adapter> \  
  -net=<network-name> \  
  -template=<template-name> \  
  <vm-name>  
  
govc vm.change  
  -vm <vm-name> \  
  -e guestinfo.userdata="$(base64 -w0 settings.toml)" \  
  -e guestinfo.userdata.encoding="base64"  
  
govc vm.power -on <vm-name>
```

Pada bagian di atas, `-e guestinfo.userdata.encoding="base64"` opsi menentukan bahwa data pengguna dikodekan base64. `-e guestinfo.userdata` Opsi ini meneruskan konten `settings.toml` berkas yang dikodekan base64 sebagai data pengguna ke instance Bottlerocket. Ganti placeholder dengan nilai spesifik Anda, seperti template Bottlerocket OVA dan detail jaringan.

Langkah 3: Verifikasi koneksi node hybrid

Setelah instans Bottlerocket dimulai, instans akan mencoba bergabung dengan cluster Amazon EKS Anda. Anda dapat memverifikasi koneksi di konsol Amazon EKS dengan menavigasi ke tab Compute untuk klaster Anda atau dengan menjalankan perintah berikut:

```
kubectl get nodes
```

 Important

Node Anda akan memiliki status `Not Ready`, yang diharapkan dan disebabkan oleh kurangnya CNI yang berjalan pada node hybrid Anda. Jika node Anda tidak bergabung dengan cluster, lihat [the section called “Pemecahan Masalah”](#).

Langkah 4: Konfigurasi CNI untuk node hybrid

Untuk membuat node hybrid Anda siap menjalankan aplikasi, lanjutkan dengan langkah-langkahnya [the section called “Konfigurasi CNI”](#).

Tingkatkan node hybrid untuk kluster Anda

Panduan untuk memutakhirkan node hibrida mirip dengan node Amazon EKS yang dikelola sendiri yang berjalan di Amazon EC2. Kami menyarankan Anda membuat node hybrid baru pada versi Kubernetes target Anda, dengan anggun memigrasikan aplikasi Anda yang ada ke node hybrid pada versi Kubernetes yang baru, dan menghapus node hybrid pada versi Kubernetes lama dari kluster Anda. Pastikan untuk meninjau [Praktik Terbaik Amazon EKS](#) untuk peningkatan sebelum memulai pemutakhiran. Amazon EKS Hybrid Nodes memiliki dukungan [versi Kubernetes yang sama untuk kluster Amazon EKS dengan node cloud](#), termasuk dukungan standar dan diperpanjang.

Amazon EKS Hybrid Nodes mengikuti [kebijakan skew versi](#) yang sama untuk node seperti Kubernetes upstream. Amazon EKS Hybrid Nodes tidak dapat menggunakan versi yang lebih baru daripada bidang kontrol Amazon EKS, dan node hybrid mungkin memiliki hingga tiga versi minor Kubernetes yang lebih lama dari versi minor bidang kontrol Amazon EKS.

Jika Anda tidak memiliki kapasitas cadangan untuk membuat node hybrid baru pada versi Kubernetes target Anda untuk strategi upgrade migrasi cutover, Anda dapat menggunakan Amazon EKS Hybrid Nodes CLI (nodeadm) untuk meng-upgrade versi Kubernetes dari node hybrid Anda di tempat.

 Important

Jika Anda memutakhirkan node hybrid Anda di tempat `nodeadm`, ada downtime untuk node selama proses di mana versi lama dari komponen Kubernetes dimatikan dan komponen versi Kubernetes yang baru diinstal dan dimulai.

Prasyarat

Sebelum memutakhirkan, pastikan Anda telah menyelesaikan prasyarat berikut.

- Versi Kubernetes target untuk upgrade node hybrid Anda harus sama atau kurang dari versi control plane Amazon EKS.
- Jika Anda mengikuti strategi upgrade migrasi cutover, node hybrid baru yang Anda instal pada versi Kubernetes target Anda harus memenuhi persyaratan. [the section called “Prasyarat”](#) Ini termasuk memiliki alamat IP dalam CIDR Jaringan Node Jarak Jauh yang Anda lewati selama pembuatan klaster Amazon EKS.
- Untuk migrasi cutover dan peningkatan di tempat, node hybrid harus memiliki akses ke [domain yang diperlukan](#) untuk menarik versi baru dari dependensi node hybrid.
- Anda harus menginstal kubectl di mesin lokal atau instance yang Anda gunakan untuk berinteraksi dengan endpoint API Amazon EKS Kubernetes Anda.
- Versi CNI Anda harus mendukung versi Kubernetes yang sedang Anda upgrade. Jika tidak, tingkatkan versi CNI Anda sebelum memutakhirkan node hybrid Anda. Untuk informasi selengkapnya, lihat [the section called “Konfigurasi CNI”](#).

Peningkatan migrasi cutover (biru-hijau)

Upgrade migrasi cutover mengacu pada proses pembuatan node hybrid baru pada host baru dengan versi Kubernetes target Anda, dengan anggun memigrasikan aplikasi Anda yang ada ke node hybrid baru pada versi Kubernetes target Anda, dan menghapus node hybrid pada versi Kubernetes lama dari klaster Anda. Strategi ini juga disebut migrasi biru-hijau.

1. Hubungkan host baru Anda sebagai node hibrida mengikuti [the section called “Connect node hybrid”](#) langkah-langkahnya. Saat menjalankan `nodeadm install` perintah, gunakan versi Kubernetes target Anda.
2. Aktifkan komunikasi antara node hybrid baru pada versi Kubernetes target dan node hybrid Anda pada versi Kubernetes lama. Konfigurasi ini memungkinkan pod untuk berkomunikasi satu sama lain saat Anda memigrasikan beban kerja Anda ke node hibrida pada versi Kubernetes target.
3. Konfirmasikan node hibrida Anda pada versi Kubernetes target Anda berhasil bergabung dengan klaster Anda dan memiliki status Siap.
4. Gunakan perintah berikut untuk menandai setiap node yang ingin Anda hapus sebagai tidak dapat dijadwalkan. Hal ini agar pod baru tidak dijadwalkan atau dijadwal ulang pada node yang

Anda ganti. Untuk informasi selengkapnya, lihat [kubectl cordon](#) di dokumentasi Kubernetes. Ganti `NODE_NAME` dengan nama node hybrid pada versi Kubernetes lama.

```
kubectl cordon NODE_NAME
```

Anda dapat mengidentifikasi dan mengikat semua node dari versi Kubernetes tertentu (dalam hal ini, 1.28) dengan cuplikan kode berikut.

```
K8S_VERSION=1.28
for node in $(kubectl get nodes -o json | jq --arg K8S_VERSION
"$K8S_VERSION" -r '.items[] | select(.status.nodeInfo.kubeletVersion |
match("\($K8S_VERSION)")).metadata.name')
do
    echo "Cordoning $node"
    kubectl cordon $node
done
```

5. Jika penerapan Anda saat ini menjalankan kurang dari dua replika CoreDNS pada node hybrid Anda, skala penerapan menjadi setidaknya dua replika. Kami menyarankan Anda menjalankan setidaknya dua replika CoreDNS pada node hybrid untuk ketahanan selama operasi normal.

```
kubectl scale deployments/coredns --replicas=2 -n kube-system
```

6. Kuras setiap node hybrid pada versi Kubernetes lama yang ingin Anda hapus dari cluster Anda dengan perintah berikut. Untuk informasi selengkapnya tentang menguras node, lihat [Safely Drain a Node](#) dalam dokumentasi Kubernetes. Ganti `NODE_NAME` dengan nama node hybrid pada versi Kubernetes lama.

```
kubectl drain NODE_NAME --ignore-daemonsets --delete-emptydir-data
```

Anda dapat mengidentifikasi dan menguras semua node dari versi Kubernetes tertentu (dalam hal ini, 1.28) dengan cuplikan kode berikut.

```
K8S_VERSION=1.28
for node in $(kubectl get nodes -o json | jq --arg K8S_VERSION
"$K8S_VERSION" -r '.items[] | select(.status.nodeInfo.kubeletVersion |
match("\($K8S_VERSION)")).metadata.name')
do
    echo "Draining $node"
```

```
kubectl drain $node --ignore-daemonsets --delete-emptydir-data
done
```

7. Anda dapat menggunakan nodeadm untuk menghentikan dan menghapus artefak node hybrid dari host. Anda harus menjalankan nodeadm dengan pengguna yang memiliki root/sudo hak istimewa. Secara default, tidak nodeadm uninstall akan melanjutkan jika ada pod yang tersisa di node. Untuk mengetahui informasi selengkapnya, lihat [the section called “Node hibrida nodeadm”](#).

```
nodeadm uninstall
```

8. Dengan artefak node hybrid dihentikan dan dihapus instalasinya, hapus sumber daya node dari cluster Anda.

```
kubectl delete node node-name
```

Anda dapat mengidentifikasi dan menghapus semua node dari versi Kubernetes tertentu (dalam hal ini, 1.28) dengan cuplikan kode berikut.

```
K8S_VERSION=1.28
for node in $(kubectl get nodes -o json | jq --arg K8S_VERSION
"$K8S_VERSION" -r '.items[] | select(.status.nodeInfo.kubeletVersion |
match("\($K8S_VERSION)").metadata.name')
do
    echo "Deleting $node"
    kubectl delete node $node
done
```

9. Tergantung pada pilihan CNI Anda, mungkin ada artefak yang tersisa di node hybrid Anda setelah menjalankan langkah-langkah di atas. Untuk informasi selengkapnya, lihat [the section called “Konfigurasi CNI”](#).

Peningkatan di tempat

Proses upgrade di tempat mengacu pada penggunaan nodeadm upgrade untuk meng-upgrade versi Kubernetes untuk node hybrid tanpa menggunakan host fisik atau virtual baru dan strategi migrasi cutover. nodeadm upgradeProses ini mematikan komponen Kubernetes lama yang sudah ada yang berjalan pada node hybrid, menghapus instalasi komponen Kubernetes lama yang sudah ada, menginstal komponen Kubernetes target baru, dan memulai komponen Kubernetes target baru. Sangat disarankan untuk memutakhirkan satu node pada satu waktu untuk meminimalkan dampak

pada aplikasi yang berjalan pada node hybrid. Durasi proses ini tergantung pada bandwidth dan latensi jaringan Anda.

1. Gunakan perintah berikut untuk menandai node yang Anda upgrade sebagai tidak dapat dijadwalkan. Hal ini agar pod baru tidak dijadwalkan atau dijadwal ulang pada node yang sedang Anda upgrade. Untuk informasi selengkapnya, lihat [kubectl cordon](#) di dokumentasi Kubernetes. Ganti NODE_NAME dengan nama node hybrid yang Anda upgrade

```
kubectl cordon NODE_NAME
```

2. Kuras node yang Anda upgrade dengan perintah berikut. Untuk informasi selengkapnya tentang menguras node, lihat [Safely Drain a Node](#) dalam dokumentasi Kubernetes. Ganti NODE_NAME dengan nama node hybrid yang Anda upgrade.

```
kubectl drain NODE_NAME --ignore-daemonsets --delete-emptydir-data
```

3. Jalankan `nodeadm upgrade` pada node hybrid yang Anda upgrade. Anda harus menjalankan `nodeadm` dengan pengguna yang memiliki root/sudo hak istimewa. Nama node dipertahankan melalui peningkatan untuk penyedia kredensi AWS SSM dan AWS IAM Roles Anywhere. Anda tidak dapat mengubah penyedia kredensi selama proses pemutakhiran. Lihat [the section called “Node hibrida nodeadm”](#) untuk nilai konfigurasi untuk `nodeConfig.yaml`. Ganti K8S_VERSION dengan versi Kubernetes target yang Anda tingkatkan.

```
nodeadm upgrade K8S_VERSION -c file://nodeConfig.yaml
```

4. Untuk memungkinkan pod dijadwalkan pada node setelah Anda memutakhirkan, ketik berikut ini. Ganti NODE_NAME dengan nama node.

```
kubectl uncordon NODE_NAME
```

5. Perhatikan status node hybrid Anda dan tunggu node Anda dimatikan dan restart pada versi Kubernetes baru dengan status Ready.

```
kubectl get nodes -o wide -w
```

Pembaruan keamanan patch untuk node hybrid

Topik ini menjelaskan prosedur untuk melakukan penambalan pembaruan keamanan di tempat untuk paket dan dependensi tertentu yang berjalan pada node hybrid Anda. Sebagai praktik terbaik, kami menyarankan Anda untuk memperbarui node hybrid Anda secara teratur untuk menerima CVEs dan patch keamanan.

Untuk langkah-langkah untuk meng-upgrade versi Kubernetes, lihat [the section called “Tingkatkan node hibrida”](#)

Salah satu contoh perangkat lunak yang mungkin memerlukan patch keamanan adalah `containerd`.

Containerd

`containerd` adalah runtime container Kubernetes standar dan dependensi inti untuk EKS Hybrid Nodes, yang digunakan untuk mengelola siklus hidup kontainer, termasuk menarik gambar dan mengelola eksekusi container. Pada node hybrid, Anda dapat menginstal `containerd` melalui [CLI `nodeadm`](#) atau secara manual. Tergantung pada sistem operasi node Anda, `nodeadm` akan menginstal `containerd` dari paket OS-distributed atau paket Docker.

Ketika CVE `containerd` telah diterbitkan, Anda memiliki opsi berikut untuk meningkatkan ke versi patch `containerd` pada node Hybrid Anda.

Langkah 1: Periksa apakah patch dipublikasikan ke manajer paket

Anda dapat memeriksa apakah patch `containerd` CVE telah dipublikasikan ke masing-masing manajer paket OS dengan mengacu pada buletin keamanan yang sesuai:

- [Amazon Linux 2023](#)
- [RHEL](#)
- [Ubuntu 20.04](#)
- [Ubuntu 22.04](#)
- [Ubuntu 24.04](#)

Jika Anda menggunakan repo Docker sebagai sumbernya `containerd`, Anda dapat memeriksa [pengumuman keamanan Docker](#) untuk mengidentifikasi ketersediaan versi yang ditambal di repo Docker.

Langkah 2: Pilih metode untuk menginstal tambalan

Ada tiga metode untuk menambal dan menginstal peningkatan keamanan di tempat pada node. Metode mana yang dapat Anda gunakan tergantung pada apakah tambalan tersedia dari sistem operasi di manajer paket atau tidak:

1. Instal tambalan dengan `nodeadm upgrade` yang dipublikasikan ke manajer paket, lihat [Langkah 2 a.](#)
2. Instal tambalan dengan manajer paket secara langsung, lihat [Langkah 2 b.](#)
3. Instal patch kustom yang tidak dipublikasikan di manajer paket. Perhatikan bahwa ada pertimbangan khusus untuk tambalan khusus untuk `containerd`, [Langkah 2 c.](#)

Langkah 2 a: Menambal dengan **nodeadm upgrade**

Setelah Anda mengonfirmasi bahwa tambalan `containerd` CVE telah dipublikasikan ke OS atau repo Docker (baik Apt atau RPM), Anda dapat menggunakan `nodeadm upgrade` perintah untuk meningkatkan ke versi terbaru. `containerd` Karena ini bukan upgrade versi Kubernetes, Anda harus meneruskan versi Kubernetes Anda saat ini ke perintah `upgrade`. `nodeadm`

```
nodeadm upgrade K8S_VERSION --config-source file:///root/nodeConfig.yaml
```

Langkah 2 b: Menambal dengan manajer paket sistem operasi

Atau Anda juga dapat memperbarui melalui manajer paket masing-masing dan menggunakannya untuk meningkatkan `containerd` paket sebagai berikut.

Amazon Linux 2023

```
sudo yum update -y
sudo yum install -y containerd
```

RHEL

```
sudo yum install -y yum-utils
sudo yum-config-manager --add-repo https://download.docker.com/linux/rhel/docker-
ce.repo
sudo yum update -y
sudo yum install -y containerd
```

Ubuntu

```
sudo mkdir -p /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/
docker.asc
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt update -y
sudo apt install -y --only-upgrade containerd.io
```

Langkah 2 c: Patch **Containerd** CVE tidak dipublikasikan di manajer paket

Jika containerd versi yang ditambah hanya tersedia dengan cara lain alih-alih di manajer paket, misalnya dalam GitHub rilis, maka Anda dapat menginstal containerd dari GitHub situs resmi.

1. Jika mesin telah bergabung dengan cluster sebagai node hybrid, maka Anda perlu menjalankan `nodeadm uninstall` perintah.
2. Instal containerd binari resmi. Anda dapat menggunakan langkah-langkah [langkah-langkah instalasi resmi](#) pada GitHub.
3. Jalankan `nodeadm install` perintah dengan `--containerd-source` argumen yang disetel ke `none`, yang akan melewati containerd instalasi `nodeadm`. Anda dapat menggunakan nilai `none` dalam containerd sumber untuk sistem operasi apa pun yang sedang dijalankan oleh node.

```
nodeadm install K8S_VERSION --credential-provider CREDS_PROVIDER --containerd-source
none
```

Hapus node hybrid

Topik ini menjelaskan cara menghapus node hybrid dari cluster Amazon EKS Anda. [Anda harus menghapus node hybrid Anda dengan alat yang kompatibel dengan Kubernetes seperti kubectl.](#)

Biaya untuk node hybrid berhenti ketika objek node dihapus dari cluster Amazon EKS. Untuk informasi selengkapnya tentang harga node hibrida, lihat [Harga Amazon EKS](#).

 Important

Menghapus node mengganggu beban kerja yang berjalan pada node. Sebelum menghapus node hybrid, sebaiknya Anda menguras node terlebih dahulu untuk memindahkan pod ke node aktif lainnya. Untuk informasi selengkapnya tentang menguras node, lihat [Safely Drain a Node](#) dalam dokumentasi Kubernetes.

Jalankan langkah-langkah kubectl di bawah ini dari mesin lokal atau instance yang Anda gunakan untuk berinteraksi dengan titik akhir API Kubernetes klaster Amazon EKS. Jika Anda menggunakan kubeconfig file tertentu, gunakan `--kubeconfig` bendera.

Langkah 1: Daftar node Anda

```
kubectl get nodes
```

Langkah 2: Tiriskan node Anda

Lihat [kubectl drain](#) di dokumentasi Kubernetes untuk informasi selengkapnya tentang perintah.

```
kubectl drain
```

```
kubectl drain --ignore-daemonsets <node-name>
```

Langkah 3: Hentikan dan hapus instalasi artefak node hybrid

Anda dapat menggunakan Amazon EKS Hybrid Nodes CLI (nodeadm) untuk menghentikan dan menghapus artefak node hybrid dari host. Anda harus menjalankan nodeadm dengan pengguna yang memiliki root/sudo hak istimewa. Secara default, tidak nodeadm `uninstall` akan melanjutkan jika ada pod yang tersisa di node. Jika Anda menggunakan AWS Systems Manager (SSM) sebagai penyedia kredensialnya, nodeadm `uninstall` perintah tersebut membatalkan pendaftaran host sebagai instans terkelola SSM. AWS Untuk informasi selengkapnya, lihat [the section called “Node hibrida nodeadm”](#).

```
nodeadm uninstall
```

Langkah 4: Hapus node Anda dari cluster

Dengan artefak node hybrid dihentikan dan dihapus, hapus sumber daya node dari cluster Anda.

```
kubectl delete node <node-name>
```

Langkah 5: Periksa artefak yang tersisa

Tergantung pada pilihan CNI Anda, mungkin ada artefak yang tersisa di node hybrid Anda setelah menjalankan langkah-langkah di atas. Lihat [the section called “Konfigurasi CNI”](#) untuk informasi selengkapnya.

Konfigurasi CNI, add-on, dan webhook untuk node hybrid

Setelah Anda mengonfigurasi cluster untuk node hybrid, node Hybrid memerlukan konfigurasi untuk Container Network Interfaces (CNI), add-on, webhook, dan pengaturan proxy yang mungkin. Untuk daftar lengkap EKS dan add-on komunitas yang kompatibel dengan node hibrida, lihat [the section called “Konfigurasi add-on”](#).

Wawasan kluster EKS EKS menyertakan pemeriksaan wawasan untuk kesalahan konfigurasi dalam penyiapan Node Hibrid EKS Anda yang dapat mengganggu fungsionalitas kluster atau beban kerja Anda. Untuk informasi lebih lanjut tentang wawasan cluster, lihat [the section called “Wawasan cluster”](#).

Berikut ini mencantumkan add-on dan komponen lain yang memiliki konfigurasi berbeda untuk node hybrid:

- Container Networking Interface (CNI): Kemampuan inti [Cilium](#) dan [Calico](#) didukung untuk digunakan dengan node hybrid. Anda dapat mengelola CNI Anda pada node hybrid dengan pilihan perkakas Anda seperti Helm. AWS VPC CNI tidak dapat digunakan dengan node hybrid. Untuk informasi selengkapnya, lihat [the section called “Konfigurasi CNI”](#).
- CoreDNS dan **kube-proxy**: CoreDNS dan diinstal secara otomatis ketika node hybrid kube-proxy bergabung dengan cluster EKS. Add-on ini dapat dikelola sebagai add-on EKS setelah pembuatan cluster.
- Ingress dan Load Balancing: Anda dapat menggunakan Load AWS Balancer Controller dan Application Load Balancer (ALB) atau Network Load Balancer (NLB) dengan ip tipe target untuk beban kerja pada node hybrid yang terhubung dengan Direct Connect atau VPN. AWS Site-to-Site Sebagai alternatif, Anda dapat menggunakan pengontrol Ingress atau penyeimbang beban pilihan Anda untuk lalu lintas aplikasi yang tetap lokal ke lingkungan lokal Anda.
- Metrik: Anda dapat menggunakan Amazon Managed Service for Prometheus (AMP) tanpa agen scraper, Distro for Open Telemetry (ADOT) AWS , dan Amazon Observability Agent dengan node

hybrid. CloudWatch Untuk menggunakan scraper tanpa agen AMP untuk metrik pod pada node hybrid, pod Anda harus dapat diakses dari VPC yang Anda gunakan untuk cluster EKS.

- Log: Anda dapat mengaktifkan pencatatan bidang kontrol EKS untuk cluster berkemampuan node hibrida. Anda dapat menggunakan add-on ADOT EKS dan add-on Amazon CloudWatch Observability Agent EKS untuk node hybrid dan pod logging.
- Identitas Pod dan IRSA: Anda dapat menggunakan EKS Pod Identities dan IAM Roles for Service Accounts (IRSA) dengan aplikasi yang berjalan pada node hybrid untuk mengaktifkan akses granular untuk pod Anda yang berjalan pada node hybrid dengan layanan lain. AWS
- Webhooks: Jika Anda menjalankan webhook, lihat pertimbangan dan langkah-langkah [the section called “Konfigurasi webhook”](#) untuk menjalankan webhook secara opsional di node cloud jika Anda tidak dapat membuat jaringan pod lokal dapat dirutekan.
- Proxy: Jika Anda menggunakan server proxy di lingkungan lokal untuk lalu lintas yang meninggalkan pusat data atau lingkungan edge, Anda dapat mengonfigurasi node dan cluster hibrida untuk menggunakan server proxy Anda. Lihat informasi yang lebih lengkap di [the section called “Konfigurasi proxy”](#).

Topik

- [Konfigurasi CNI untuk node hybrid](#)
- [Konfigurasi add-on untuk node hybrid](#)
- [Konfigurasi webhook untuk node hybrid](#)
- [Konfigurasi proxy untuk node hybrid](#)

Konfigurasi CNI untuk node hybrid

Cilium dan Calico didukung sebagai Container Networking Interfaces (CNI) untuk Amazon EKS Hybrid Nodes. Anda harus menginstal CNI untuk node hybrid agar siap melayani beban kerja. Node hibrida muncul dengan status Not Ready sampai CNI berjalan. Anda dapat mengelola ini CNI dengan alat pilihan Anda seperti Helm. Amazon VPC CNI tidak kompatibel dengan node hybrid dan VPC CNI dikonfigurasi dengan anti-afinitas untuk label. `eks.amazonaws.com/compute-type: hybrid`

Kompatibilitas versi CNI

Versi v1.16.x cilium didukung dan direkomendasikan untuk EKS Hybrid Nodes untuk setiap versi Kubernetes yang didukung di Amazon EKS.

Versi v3.29.x Calico didukung dan direkomendasikan untuk EKS Hybrid Nodes untuk setiap versi Kubernetes yang didukung di Amazon EKS.

Lihat [dukungan versi Kubernetes untuk versi](#) Kubernetes yang didukung oleh Amazon EKS. EKS Hybrid Nodes memiliki dukungan versi Kubernetes yang sama dengan cluster Amazon EKS dengan node cloud.

Kemampuan yang didukung

AWS memberikan dukungan teknis untuk kemampuan Cilium dan Calico berikut untuk digunakan dengan node hibrida. Jika Anda berencana untuk menggunakan fungsionalitas di luar lingkup AWS dukungan, kami sarankan Anda mendapatkan dukungan komersial untuk plugin atau memiliki keahlian internal untuk memecahkan masalah dan berkontribusi perbaikan pada proyek plugin CNI.

Fitur	Cilium	Calico
Kesesuaian jaringan Kubernetes	Ya	Ya
Kontrol bidang ke konektivitas simpul	Ya	Ya
Kontrol pesawat ke konektivitas pod	Ya	Ya
Manajemen Siklus Hidup	Instal, Tingkatkan, Hapus	Instal, Tingkatkan, Hapus
Mode Jaringan	VXLAN	VXLAN
Manajemen Alamat IP (IPAM)	Lingkup Cluster (Cilium IPAM)	Calico IPAM
Keluarga IP	IPv4	IPv4
BGP	Ya (Pesawat Kontrol Cilium)	Ya

Pertimbangan cilium

- [Secara default, Cilium dikonfigurasi untuk berjalan dalam mode overlay/tunnel dengan VXLAN sebagai metode enkapsulasi.](#) Mode ini memiliki persyaratan paling sedikit pada jaringan fisik yang mendasarinya.

- Secara default, Cilium [menyamarkan](#) alamat IP sumber dari semua lalu lintas pod meninggalkan cluster ke alamat IP node. Hal ini memungkinkan untuk menjalankan Cilium dengan node hybrid, terlepas dari apakah jaringan pod jarak jauh dikonfigurasi pada cluster atau tidak. Jika Anda menonaktifkan masquerading, pod Anda CIDRs harus dapat dirutekan di jaringan lokal dan Anda harus mengonfigurasi klaster Amazon EKS dengan jaringan pod jarak jauh.
- Jika Anda menjalankan webhook pada node hybrid, pod Anda CIDRs harus dapat dirutekan di jaringan lokal dan Anda harus mengonfigurasi klaster Amazon EKS dengan jaringan pod jarak jauh. Jika pod CIDRs Anda tidak dapat dirutekan di jaringan lokal, sebaiknya jalankan webhook di node cloud di cluster yang sama. Lihat [Mengonfigurasi webhook untuk node hybrid untuk informasi selengkapnya](#).
- Cara umum untuk membuat pod CIDR dapat dirutekan di jaringan lokal adalah dengan mengiklankan alamat pod dengan BGP. Untuk menggunakan BGP dengan Cilium, Anda harus mengatur `bgpControlPlane.enabled: true` konfigurasi Helm Anda. Untuk informasi lebih lanjut tentang dukungan BGP Cilium, lihat Cilium BGP Control Plane dalam [dokumentasi Cilium](#).
- IP Address Management (IPAM) default di Cilium disebut [Cluster Scope](#), di mana operator Cilium mengalokasikan alamat IP untuk setiap node berdasarkan pod yang dikonfigurasi pengguna. CIDRs Pod CIDRs dikonfigurasi dengan nilai `clusterPoolIPv4PodCIDRList` Helm, yang seharusnya cocok dengan jaringan pod jarak jauh yang CIDRs Anda konfigurasikan untuk klaster Amazon EKS Anda. Cilium mengalokasikan segmen dari ke setiap node. `clusterPoolIPv4PodCIDRList` Ukuran segmen per node dikonfigurasi dengan nilai `clusterPoolIPv4MaskSize` Helm. Untuk informasi selengkapnya tentang `clusterPoolIPv4PodCIDRList` dan `clusterPoolIPv4MaskSize`, lihat [Memperluas kumpulan cluster](#) dalam dokumentasi Cilium.

Instal Cilium pada node hybrid

1. Pastikan Anda telah menginstal Helm CLI di lingkungan baris perintah Anda. Lihat [helm penyiapan](#) untuk petunjuk penginstalan.
2. Instal repo Cilium Helm.

```
helm repo add cilium https://helm.cilium.io/
```

3. Buat file YAMM bernama `cilium-values.yaml`. Contoh berikut mengkonfigurasi Cilium untuk berjalan hanya pada node hybrid dengan menetapkan afinitas untuk label `eks.amazonaws.com/compute-type: hybrid`

- Jika Anda mengonfigurasi cluster Amazon EKS Anda dengan jaringan pod jarak jauh, konfigurasi pod yang sama CIDRs untuk `AndaclusterPoolIPv4PodCIDRList`. Misalnya, `10.100.0.0/24`. CIDR pod lokal Anda tidak boleh tumpang tindih dengan CIDR node lokal Anda saat menjalankan CNI dalam mode overlay/tunnel.
- Konfigurasi `clusterPoolIPv4MaskSize` berdasarkan pod yang Anda butuhkan per node. Misalnya, 25 untuk ukuran segmen /25 128 pod per node.
- Anda tidak boleh mengubah `clusterPoolIPv4PodCIDRList` atau `clusterPoolIPv4MaskSize` setelah menerapkan Cilium di kluster Anda, lihat [Memperluas kumpulan cluster di dokumentasi Cilium](#).
- Untuk daftar lengkap nilai Helm untuk Cilium, lihat [referensi Helm](#) di dokumentasi Cilium.

```

affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
        - matchExpressions:
            - key: eks.amazonaws.com/compute-type
              operator: In
              values:
                - hybrid
ipam:
  mode: cluster-pool
  operator:
    clusterPoolIPv4MaskSize: 25
    clusterPoolIPv4PodCIDRList:
      - POD_CIDR
loadBalancer:
  serviceTopology: true
operator:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
          - matchExpressions:
              - key: eks.amazonaws.com/compute-type
                operator: In
                values:
                  - hybrid
  unmanagedPodWatcher:
    restart: false
envoy:

```

```
enabled: false
```

4. Instal Cilium di cluster Anda.

- Ganti `CILIUM_VERSION` dengan versi Cilium (misalnya `v1.17.0`). Kami menyarankan Anda menjalankan versi patch terbaru untuk versi minor Cilium Anda. Anda dapat menemukan rilis patch terbaru untuk rilis Cilium minor tertentu di [bagian Stable Releases](#) dari dokumentasi Cilium.
- Jika Anda mengaktifkan BGP untuk penerapan Anda, tambahkan `--set bgpControlPlane.enabled=true` bendera pada perintah di bawah ini.
- Jika Anda menggunakan file kubeconfig tertentu, gunakan `--kubeconfig` flag dengan perintah Helm install.

```
helm install cilium cilium/cilium \
  --version CILIUM_VERSION \
  --namespace kube-system \
  --values cilium-values.yaml
```

5. Anda dapat mengonfirmasi instalasi Cilium Anda berhasil dengan perintah berikut. Anda akan melihat `cilium-operator` penerapan dan `cilium-agent` berjalan pada setiap node hybrid Anda. Selain itu, node hybrid Anda sekarang harus memiliki `statusReady`. Untuk informasi tentang cara mengkonfigurasi BGP untuk Cilium, lanjutkan ke langkah berikutnya.

```
kubectl get pods -n kube-system
```

NAME	READY	STATUS	RESTARTS	AGE
cilium-jjjn8	1/1	Running	0	11m
cilium-operator-d4f4d7fcb-sc5xn	1/1	Running	0	11m

```
kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
mi-04a2cf999b7112233	Ready	<none>	19m	v1.31.0-eks-a737599

6. Untuk menggunakan BGP dengan Cilium untuk mengiklankan alamat pod Anda dengan jaringan lokal, Anda harus menginstal Cilium dengan `bgpControlPlane.enabled: true`. Jika Anda mengaktifkan BGP untuk penerapan Cilium yang ada, Anda harus memulai ulang operator Cilium untuk menerapkan konfigurasi BGP jika BGP sebelumnya tidak diaktifkan. Anda dapat mengatur

`operator.rollOutPods` ke `true` dalam nilai Helm Anda untuk memulai ulang operator Cilium sebagai bagian dari proses instal/pemutakhiran Helm.

Untuk mengkonfigurasi BGP di Cilium, pertama buat file yang disebut `cilium-bgp-cluster.yaml` dengan definisi `CiliumBGPClusterConfig` Anda mungkin perlu mendapatkan informasi berikut dari administrator jaringan Anda.

- Konfigurasikan `localASN` dengan ASN untuk node yang menjalankan Cilium.
- Konfigurasikan `peerASN` dengan ASN untuk router lokal Anda.
- Konfigurasikan `peerAddress` dengan IP router lokal yang akan dilayani oleh setiap node yang menjalankan Cilium.

```
apiVersion: cilium.io/v2alpha1
kind: CiliumBGPClusterConfig
metadata:
  name: cilium-bgp
spec:
  nodeSelector:
    matchExpressions:
      - key: eks.amazonaws.com/compute-type
        operator: In
        values:
          - hybrid
  bgpInstances:
    - name: "rack0"
      localASN: NODES_ASN
      peers:
        - name: "onprem-router"
          peerASN: ONPREM_ROUTER_ASN
          peerAddress: ONPREM_ROUTER_IP
          peerConfigRef:
            name: "cilium-peer"
```

7. Terapkan konfigurasi Cilium BGP Cluster ke cluster Anda.

```
kubectl apply -f cilium-bgp-cluster.yaml
```

8. Sumber `CiliumBGPPeerConfig` daya mendefinisikan konfigurasi rekan BGP. Beberapa rekan dapat berbagi konfigurasi yang sama dan memberikan referensi ke `CiliumBGPPeerConfig` sumber daya bersama. Buat file bernama `cilium-bgp-peer.yaml` untuk mengonfigurasi

konfigurasi rekan untuk jaringan lokal Anda. Lihat [Konfigurasi Peer BGP](#) di dokumentasi Cilium untuk daftar lengkap opsi konfigurasi.

```
apiVersion: cilium.io/v2alpha1
kind: CiliumBGPPeerConfig
metadata:
  name: cilium-peer
spec:
  timers:
    holdTimeSeconds: 30
    keepAliveTimeSeconds: 10
  gracefulRestart:
    enabled: true
    restartTimeSeconds: 120
  families:
    - afi: ipv4
      safi: unicast
      advertisements:
        matchLabels:
          advertise: "bgp"
```

9. Terapkan konfigurasi Cilium BGP Peer ke cluster Anda.

```
kubectl apply -f cilium-bgp-peer.yaml
```

10. `CiliumBGPAdvertisement` sumber daya digunakan untuk menentukan berbagai jenis iklan dan atribut yang terkait dengannya. Buat file bernama `cilium-bgp-advertisement.yaml` dan konfigurasi `CiliumBGPAdvertisement` sumber daya dengan pengaturan Anda.

```
apiVersion: cilium.io/v2alpha1
kind: CiliumBGPAdvertisement
metadata:
  name: bgp-advertisements
  labels:
    advertise: bgp
spec:
  advertisements:
    - advertisementType: "PodCIDR"
    - advertisementType: "Service"
  service:
    addresses:
      - ClusterIP
```

- ExternalIP
- LoadBalancerIP

11. Terapkan konfigurasi Cilium BGP Advertisement ke cluster Anda.

```
kubectl apply -f cilium-bgp-advertisement.yaml
```

Anda dapat mengonfirmasi bahwa peering BGP bekerja dengan Cilium [CLI](#) dengan menggunakan perintah `cilium bgp peers`. Anda akan melihat nilai yang benar dalam output untuk lingkungan Anda dan Status Sesi sebagai `established`. Lihat [Panduan Pemecahan Masalah dan Operasi](#) di dokumentasi Cilium untuk informasi selengkapnya tentang pemecahan masalah.

Tingkatkan Cilium pada node hibrida

Sebelum memutakhirkan penerapan Cilium Anda, tinjau [dokumentasi pemutakhiran Cilium dan catatan pemutakhiran](#) dengan cermat untuk memahami perubahan dalam versi Cilium target.

1. Pastikan Anda telah menginstal helm CLI di lingkungan baris perintah Anda. Lihat [dokumentasi Helm](#) untuk petunjuk penginstalan.
2. Instal repo Cilium Helm.

```
helm repo add cilium https://helm.cilium.io/
```

3. Jalankan pemeriksaan pra-penerbangan peningkatan Cilium. Ganti `CILIUM_VERSION` dengan versi Cilium target Anda. Kami menyarankan Anda menjalankan versi patch terbaru untuk versi minor Cilium Anda. Anda dapat menemukan rilis patch terbaru untuk rilis Cilium minor tertentu di [bagian Stable Releases](#) dari dokumentasi Cilium.

```
helm install cilium-preflight cilium/cilium --version CILIUM_VERSION \
  --namespace=kube-system \
  --set preflight.enabled=true \
  --set agent=false \
  --set operator.enabled=false
```

4. Setelah menerapkan `cilium-preflight.yaml`, pastikan jumlah pod sama dengan jumlah READY pod Cilium yang berjalan.

```
kubectl get ds -n kube-system | sed -n '1p;/cilium/p'
```

NAME	DESIRED	CURRENT	READY	UP-TO-DATE	AVAILABLE	NODE
SELECTOR						
AGE						
cilium	2	2	2	2	2	<none>
1h20m						
cilium-pre-flight-check	2	2	2	2	2	<none>
7m15s						

5. Setelah jumlah pod READY sama, pastikan penerapan pra-penerbangan Cilium juga ditandai sebagai READY 1/1. Jika menunjukkan READY 0/1, lihat bagian [Validasi CNP](#) dan selesaikan masalah dengan penerapan sebelum melanjutkan dengan peningkatan.

```
kubectl get deployment -n kube-system cilium-pre-flight-check -w
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
cilium-pre-flight-check	1/1	1	0	12s

6. Hapus preflight

```
helm uninstall cilium-preflight --namespace kube-system
```

7. Selama operasi cluster normal, semua komponen Cilium harus menjalankan versi yang sama. Langkah-langkah berikut menjelaskan cara memutakhirkan semua komponen dari satu rilis stabil ke rilis stabil selanjutnya. Saat memutakhirkan dari satu rilis minor ke rilis minor lainnya, sebaiknya Anda meng-upgrade ke rilis patch terbaru untuk versi minor Cilium yang ada terlebih dahulu. Untuk meminimalkan gangguan, atur `upgradeCompatibility` opsi ke versi Cilium awal yang Anda instal di cluster ini.

Sebelum menjalankan perintah helm upgrade, pertahankan nilai untuk penerapan Anda di `cilium-values.yaml` atau gunakan opsi baris `--set` perintah untuk pengaturan Anda. Operasi pemutakhiran menimpa Cilium ConfigMap, jadi sangat penting bahwa nilai konfigurasi Anda diteruskan saat Anda memutakhirkan. Jika Anda menggunakan BGP, kami sarankan Anda menggunakan opsi baris `--set bgpControlPlane=true` perintah alih-alih memberikan informasi ini dalam file nilai Anda.

```
helm upgrade cilium cilium/cilium --version CILIUM_VERSION \
  --namespace kube-system \
  --set upgradeCompatibility=1.X \
  -f cilium-values.yaml
```

8. (Opsional) Jika Anda perlu mengembalikan upgrade Anda karena masalah, jalankan perintah berikut.

```
helm history cilium --namespace kube-system
helm rollback cilium [REVISION] --namespace kube-system
```

Hapus Cilium dari node hibrida

1. Jalankan perintah berikut untuk menghapus semua komponen Cilium dari cluster Anda. Catatan, menghapus instalasi CNI dapat memengaruhi kesehatan node dan pod dan tidak boleh dilakukan pada cluster produksi.

```
helm uninstall cilium --namespace kube-system
```

Antarmuka dan rute yang dikonfigurasi oleh Cilium tidak dihapus secara default saat CNI dihapus dari cluster, lihat [GitHub masalah](#) untuk informasi selengkapnya.

2. Untuk membersihkan file konfigurasi dan sumber daya on-disk, jika Anda menggunakan direktori konfigurasi standar, Anda dapat menghapus file seperti yang ditunjukkan oleh [cni-uninstall.shskrip](#) di repositori Cilium pada GitHub
3. Untuk menghapus Cilium Custom Resource Definitions (CRDs) dari cluster Anda, Anda dapat menjalankan perintah berikut.

```
kubectl get crds -oname | grep "cilium" | xargs kubectl delete
```

Pertimbangan Calico

- [Kami menyarankan Anda menjalankan Calico dalam mode overlay/tunnel dengan VXLAN sebagai metode enkapsulasi](#). Mode ini memiliki persyaratan paling sedikit pada jaringan fisik yang mendasarinya. Untuk informasi lebih lanjut tentang mode jaringan Calico yang berbeda, lihat [Menentukan opsi jaringan terbaik](#) dalam dokumentasi Calico.
- Kami menyarankan Anda menjalankan Calico dengan `natOutgoing` set `true`. Dengan `natOutgoing` set `true`, alamat IP sumber dari semua lalu lintas pod yang meninggalkan cluster diterjemahkan ke alamat IP node. Hal ini memungkinkan untuk menjalankan Calico dengan kluster Amazon EKS, terlepas dari apakah jaringan pod jarak jauh dikonfigurasi di cluster atau tidak. Jika

Anda menonaktifkan `Outgoing`, pod Anda CIDRs harus dapat dirutekan di jaringan lokal dan Anda harus mengonfigurasi kluster Amazon EKS dengan jaringan pod jarak jauh.

- Jika Anda menjalankan webhook pada node hybrid, pod Anda CIDRs harus dapat dirutekan di jaringan lokal dan Anda harus mengonfigurasi kluster Amazon EKS dengan jaringan pod jarak jauh. Jika pod CIDRs Anda tidak dapat dirutekan di jaringan lokal, sebaiknya jalankan webhook di node cloud di cluster yang sama. Lihat [Mengonfigurasi webhook untuk node hybrid untuk informasi selengkapnya](#).
- Cara umum untuk membuat pod CIDR dapat dirutekan di jaringan lokal adalah dengan mengiklankan alamat pod dengan BGP. Untuk menggunakan BGP dengan Calico, Anda harus mengatur konfigurasi `installation.calicoNetwork.bgp: Enabled` Helm Anda. Untuk informasi selengkapnya tentang dukungan BGP Calico, lihat [Mengonfigurasi peering BGP](#) di dokumentasi Calico.
- Manajemen Alamat IP default (IPAM) di Calico disebut [Calico IPAM](#), di mana `calico-ipam` plugin mengalokasikan alamat IP untuk setiap node berdasarkan pod yang dikonfigurasi pengguna. CIDRs Pod CIDRs dikonfigurasi dengan nilai `installation.calicoNetwork.ipPools.cidr` Helm, yang seharusnya cocok dengan jaringan pod jarak jauh yang CIDRs Anda konfigurasikan untuk kluster Amazon EKS Anda. Calico mengalokasikan segmen dari `ipPools.cidr` ke setiap node. Ukuran segmen per node dikonfigurasi dengan nilai `ipPools.blockSize` Helm. Untuk informasi lebih lanjut tentang IPAM dengan Calico, lihat [Memulai dengan manajemen alamat IP dalam dokumentasi](#) Calico.

Instal Calico pada node hybrid

1. Pastikan Anda telah menginstal CLI helm di lingkungan baris perintah Anda. Lihat [dokumentasi Helm](#) untuk petunjuk penginstalan.
2. Instal repo Cilium Helm.

```
helm repo add projectcalico https://docs.tigera.io/calico/charts
```

3. Buat file YAMM bernama `calico-values.yaml`. Contoh berikut mengkonfigurasi semua komponen Calico untuk berjalan hanya pada node hybrid dengan menetapkan afinitas untuk label `eks.amazonaws.com/compute-type: hybrid`
 - Ganti `POD_CIDR` dengan rentang CIDR untuk pod Anda. Jika Anda mengonfigurasi kluster Amazon EKS Anda dengan jaringan pod jarak jauh, `POD_CIDR` yang Anda tentukan untuk Calico harus sama dengan jaringan pod jarak jauh. Misalnya, `10.100.0.0/24`. CIDR pod lokal

Anda tidak boleh tumpang tindih dengan CIDR node lokal Anda saat menjalankan CNI dalam mode overlay/tunnel.

- Ganti `CIDR_SIZE` dengan ukuran segmen CIDR yang ingin Anda alokasikan ke setiap node. Misalnya, 25 untuk ukuran segmen /25 128 alamat pod per node. Untuk informasi selengkapnya tentang CIDR `blockSize` dan mengubah `blockSize`, lihat [Mengubah ukuran blok kumpulan IP](#) dalam dokumentasi Calico.
- Dalam contoh di bawah ini, `natOutgoing` diaktifkan dan `bgp` dinonaktifkan. Ubah nilai-nilai ini berdasarkan konfigurasi target Anda.

```
installation:
  enabled: true
  cni:
    type: Calico
    ipam:
      type: Calico
  calicoNetwork:
    bgp: Disabled
    ipPools:
      - cidr: POD_CIDR
        blockSize: CIDR_SIZE
        encapsulation: VXLAN
        natOutgoing: Enabled
        nodeSelector: eks.amazonaws.com/compute-type == "hybrid"
  controlPlaneReplicas: 1
  controlPlaneNodeSelector:
    eks.amazonaws.com/compute-type: hybrid
  calicoNodeDaemonSet:
    spec:
      template:
        spec:
          nodeSelector:
            eks.amazonaws.com/compute-type: hybrid
  csiNodeDriverDaemonSet:
    spec:
      template:
        spec:
          nodeSelector:
            eks.amazonaws.com/compute-type: hybrid
  calicoKubeControllersDeployment:
    spec:
      template:
        spec:
```

```

      nodeSelector:
        eks.amazonaws.com/compute-type: hybrid
  typhaDeployment:
    spec:
      template:
        spec:
          nodeSelector:
            eks.amazonaws.com/compute-type: hybrid

```

4. Instal Calico di cluster Anda.

- Ganti CALICO_VERSION dengan versi Calico (misalnya v3.29.0), lihat rilis [Calico untuk menemukan rilis](#) patch terbaru untuk versi minor Calico Anda. Kami menyarankan Anda menjalankan versi patch terbaru untuk versi minor Calico.
- Jika Anda menggunakan kubeconfig file tertentu, gunakan --kubeconfig bendera.

```

helm install calico projectcalico/tigera-operator \
  --version CALICO_VERSION \
  --namespace kube-system \
  -f calico-values.yaml

```

5. Anda dapat mengonfirmasi bahwa instalasi Calico Anda berhasil dengan perintah berikut. Anda akan melihat tigera-operator penerapan, calico-node agen berjalan pada setiap node hybrid Anda, the calico-apiservercsi-node-driver, dan calico-kube-controllers deployed. Selain itu, node hybrid Anda sekarang harus memiliki statusReady. Jika Anda menggunakan natOutgoing: Disabled, maka semua komponen Calico tidak akan dapat memulai dengan sukses sampai Anda mengiklankan alamat pod Anda dengan jaringan lokal Anda. Untuk informasi tentang cara mengkonfigurasi BGP untuk Calico, lanjutkan ke langkah berikutnya.

```
kubectl get pods -A
```

NAMESPACE	NAME	READY	STATUS	
calico-apiserver	calico-apiserver-6c77bb6d46-2n8mq	1/1	Running	0
calico-system	calico-kube-controllers-7c5f8556b5-7h267	1/1	Running	0
calico-system	calico-node-s5nnk	1/1	Running	0

calico-system 69s	calico-typha-6487cc9d8c-wc5jm	1/1	Running	0
calico-system 68s	csi-node-driver-cv42d	2/2	Running	0
kube-system 6m27s	coredns-7bb495d866-2lc9v	1/1	Running	0
kube-system 157m	coredns-7bb495d866-2t8ln	1/1	Running	0
kube-system 18m	kube-proxy-lxzxh	1/1	Running	0
kube-system 90s	tigera-operator-f8bc97d4c-28b4d	1/1	Running	0

```
kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
mi-0c6ec2f6f79176565	Ready	<none>	5h13m	v1.31.0-eks-a737599

6. Jika Anda menginstal Calico tanpa BGP, lewati langkah ini. Untuk mengkonfigurasi BGP, buat file yang disebut `calico-bgp.yaml` dengan BGPPeer konfigurasi dan file. BGPConfiguration Penting untuk membedakan BGPPeer dan BGPConfiguration. BGPPeer ini adalah router berkemampuan BGP atau sumber daya jarak jauh yang dengannya node dalam cluster Calico akan mengintip. BGPPeerKonfigurasi asNumber dalam mirip dengan pengaturan Cilium. peerASN BGPConfiguration ini diterapkan ke setiap node Calico dan asNumber untuk BGPConfiguration setara dengan pengaturan Cilium. localASN Ganti ONPREM_ROUTER_IP ONPREM_ROUTER_ASN,, dan LOCAL_ASN dalam contoh di bawah ini dengan nilai untuk lingkungan lokal Anda, yang mungkin harus Anda peroleh dari administrator jaringan. keepOriginalNextHop: true Pengaturan ini digunakan untuk memastikan setiap node hanya mengiklankan CIDR jaringan pod yang dimilikinya.

```
apiVersion: projectcalico.org/v3
kind: BGPPeer
metadata:
  name: calico-hybrid-nodes
spec:
  peerIP: ONPREM_ROUTER_IP
  asNumber: ONPREM_ROUTER_ASN
  keepOriginalNextHop: true
---
apiVersion: projectcalico.org/v3
```

```
kind: BGPConfiguration
metadata:
  name: default
spec:
  nodeToNodeMeshEnabled: false
  asNumber: LOCAL_ASN
```

7. Terapkan file ke cluster Anda.

```
kubectl apply -f calico-bgp.yaml
```

8. Konfirmasikan bahwa pod Calico berjalan dengan perintah berikut.

```
kubectl get pods -n calico-system -w
```

NAMESPACE	NAME	READY	STATUS	
RESTARTS	AGE			
calico-apiserver	calico-apiserver-598bf99b6c-2vltk 3h24m	1/1	Running	0
calico-system (59m ago)	calico-kube-controllers-75f84bbfd6-zwmnx 3h20m	1/1	Running	31
calico-system	calico-node-9b2pg 5h17m	1/1	Running	0
calico-system	calico-typha-7d55c76584-kxtnq 5h18m	1/1	Running	0
calico-system	csi-node-driver-dmnmm 5h18m	2/2	Running	0
kube-system	coredns-7bb495d866-dtn4z 6h23m	1/1	Running	0
kube-system	coredns-7bb495d866-mk7j4 6h19m	1/1	Running	0
kube-system	kube-proxy-vms28 6h12m	1/1	Running	0
kube-system	tigera-operator-55f9d9d565-jj9bg 73m	1/1	Running	0

Jika Anda mengalami masalah selama langkah-langkah ini, lihat [panduan pemecahan masalah di dokumentasi](#) Calico.

Tingkatkan Calico pada node hibrida

Sebelum memutakhirkan penerapan Calico Anda, tinjau [dokumentasi pemutakhiran Calico](#) dan [catatan rilis](#) dengan cermat untuk memahami perubahan dalam versi Calico target. Langkah-langkah peningkatan bervariasi berdasarkan apakah Anda menggunakan Helm, operator Calico, dan jenis datastore. Langkah-langkah di bawah ini mengasumsikan penggunaan Helm.

1. Unduh manifes operator untuk versi Calico yang Anda tingkatkan. Ganti `CALICO_VERSION` dengan versi yang Anda tingkatkan, misalnya `v3.29.0`. Pastikan untuk menambahkan ke `major.minor.patch`.

```
kubectl apply --server-side --force-conflicts \
  -f https://raw.githubusercontent.com/projectcalico/calico/CALICO_VERSION/
  manifests/operator-crds.yaml
```

2. Jalankan `helm upgrade` untuk meningkatkan penyebaran Calico Anda. Ganti `CALICO_VERSION` dengan versi yang Anda tingkatkan, misalnya `v3.29.0`. Buat `calico-values.yaml` file dari nilai konfigurasi yang Anda gunakan untuk menginstal Calico.

```
helm upgrade calico projectcalico/tigera-operator \
  --version CALICO_VERSION \
  --namespace kube-system \
  -f calico-values.yaml
```

Hapus Calico dari node hybrid

1. Jalankan perintah berikut untuk menghapus komponen Calico dari cluster Anda. Perhatikan bahwa menghapus instalasi CNI dapat memengaruhi kesehatan node dan pod dan tidak boleh dilakukan pada cluster produksi. Jika Anda menginstal Calico di namespace selain `kube-system` mengubah namespace dalam perintah di bawah ini.

```
helm uninstall calico --namespace kube-system
```

Perhatikan bahwa antarmuka dan rute yang dikonfigurasi oleh Calico tidak dihapus secara default saat Anda menghapus CNI dari cluster.

2. Untuk membersihkan file konfigurasi dan sumber daya pada disk, hapus file Calico dari direktori `/opt/cni` dan `/etc/cni`.
3. Untuk menghapus Calico CRDs dari cluster Anda, jalankan perintah berikut.

```
kubectl get crds -oname | grep "calico" | xargs kubectl delete
```

```
kubectl get crds -oname | grep "tigera" | xargs kubectl delete
```

Konfigurasi add-on untuk node hybrid

Halaman ini menjelaskan pertimbangan untuk menjalankan AWS add-on dan add-on komunitas di Amazon EKS Hybrid Nodes. Untuk mempelajari lebih lanjut tentang add-on Amazon EKS dan proses untuk membuat, meningkatkan, dan menghapus add-on dari kluster Anda, lihat [the section called “Add-on Amazon EKS”](#). Kecuali dinyatakan lain di halaman ini, proses untuk membuat, meningkatkan, dan menghapus add-on Amazon EKS sama untuk kluster Amazon EKS dengan node hibrida seperti halnya untuk cluster Amazon EKS dengan node yang berjalan di Cloud. AWS Hanya add-on yang disertakan pada halaman ini yang telah divalidasi untuk kompatibilitas dengan Amazon EKS Hybrid Nodes.

AWS Add-on berikut ini kompatibel dengan Amazon EKS Hybrid Nodes.

AWS add-on	Versi add-on yang kompatibel
kube-proxy	v1.25.14-eksbuild.2 dan di atas
CoreDNS	v1.9.3-eksbuild.7 dan di atas
AWS Distro untuk OpenTelemetry (ADOT)	v0.102.1-eksbuild.2 dan di atas
CloudWatch Agen observabilitas	v2.2.1-eksbuild.1 dan di atas
Agen Identitas EKS Pod	- v1.3.3-eksbuild.1 dan di atas, kecuali untuk Bottlerocket - v1.3.7-eksbuild.2 dan di atas untuk Bottlerocket
Agen pemantauan simpul	v1.2.0-eksbuild.1 dan di atas
Pengontrol snapshot CSI	v8.1.0-eksbuild.1 dan di atas
AWS Konektor CA Pribadi untuk Kubernetes	v1.6.0-eksbuild.1 dan di atas

Add-on komunitas berikut kompatibel dengan Amazon EKS Hybrid Nodes. Untuk mempelajari lebih lanjut tentang add-on komunitas, lihat [the section called “Pengaya komunitas”](#).

Add-on komunitas	Versi add-on yang kompatibel
Server Metrik Kubernetes	v0.7.2-eksbuild.1 dan di atas
manajer sertifikat	v1.17.2-eksbuild.1 dan di atas
Prometheus Node Exportir	v1.9.1-eksbuild.2 dan di atas
kube-state-metrics	v2.15.0-eksbuild.4 dan di atas

Selain add-on Amazon EKS pada tabel di atas, [Amazon Managed Service untuk Prometheus Collector](#), dan Load [AWS Balancer Controller untuk masuknya aplikasi \(HTTP\) dan load balancing \(TCP/UDP\)](#) kompatibel dengan node hybrid.

Ada AWS add-on dan add-on komunitas yang tidak kompatibel dengan Amazon EKS Hybrid Nodes. Versi terbaru dari add-on ini memiliki aturan anti-afinitas untuk `eks.amazonaws.com/compute-type: hybrid` label default yang diterapkan pada node hybrid. Ini mencegah mereka berjalan pada node hybrid saat digunakan di cluster Anda. Jika Anda memiliki cluster dengan node hybrid dan node yang berjalan di AWS Cloud, Anda dapat menerapkan add-on ini di cluster Anda ke node yang berjalan di Cloud. AWS Amazon VPC CNI tidak kompatibel dengan node hybrid, dan Cilium dan Calico didukung sebagai Container Networking Interfaces (CNI) untuk Amazon EKS Hybrid Nodes. Untuk informasi selengkapnya, lihat [the section called “Konfigurasi CNI”](#).

AWS add-on

Bagian berikut menjelaskan perbedaan antara menjalankan AWS add-on yang kompatibel pada node hybrid dibandingkan dengan jenis komputasi Amazon EKS lainnya.

kube-proxy dan CoreDNS

EKS menginstal kube-proxy dan CoreDNS sebagai add-on yang dikelola sendiri secara default saat Anda membuat cluster EKS dengan API dan, termasuk dari CLI. AWS SDKs AWS Anda dapat menimpa add-on ini dengan add-on Amazon EKS setelah pembuatan cluster. Referensi dokumentasi EKS untuk detail tentang [the section called “kube-proxy”](#) dan [the section called “CoreDNS”](#). Jika Anda menjalankan cluster mode campuran dengan node hybrid dan node di AWS Cloud, sebaiknya Anda memiliki setidaknya satu replika CoreDNS pada node hybrid dan setidaknya satu replika CoreDNS di

node Anda di Cloud. AWS Lihat [the section called “Konfigurasi replika CoreDNS”](#) langkah-langkah konfigurasi.

CloudWatch Agen observabilitas

Operator agen CloudWatch Observability menggunakan [webhook](#). Jika Anda menjalankan operator pada node hibrid, CIDR pod lokal Anda harus dapat dirutekan di jaringan lokal dan Anda harus mengonfigurasi kluster EKS Anda dengan jaringan pod jarak jauh Anda. Untuk informasi selengkapnya, lihat [Mengonfigurasi webhook untuk node hibrid](#).

Metrik tingkat simpul tidak tersedia untuk node hibrida karena [CloudWatch Wawasan Kontainer](#) bergantung pada ketersediaan [Layanan Metadata Instance \(IMDS\) untuk metrik tingkat simpul](#). Metrik tingkat cluster, beban kerja, pod, dan container tersedia untuk node hibrid.

Setelah menginstal add-on dengan mengikuti langkah-langkah yang dijelaskan dalam [Instal CloudWatch agen dengan Amazon CloudWatch Observability](#), manifes add-on harus diperbarui sebelum agen dapat berjalan dengan sukses di node hibrid. Edit `amazoncloudwatchagents` sumber daya pada cluster untuk menambahkan variabel `RUN_WITH_IRSA` lingkungan seperti yang ditunjukkan di bawah ini.

```
kubectl edit amazoncloudwatchagents -n amazon-cloudwatch cloudwatch-agent
```

```
apiVersion: v1
items:
- apiVersion: cloudwatch.aws.amazon.com/v1alpha1
  kind: AmazonCloudWatchAgent
  metadata:
    ...
    name: cloudwatch-agent
    namespace: amazon-cloudwatch
    ...
  spec:
    ...
    env:
    - name: RUN_WITH_IRSA # <-- Add this
      value: "True" # <-- Add this
    - name: K8S_NODE_NAME
      valueFrom:
        fieldRef:
          fieldPath: spec.nodeName
    ...
```

Kolektor terkelola Prometheus yang Dikelola Amazon untuk node hibrida

Kolektor terkelola Amazon Managed Service for Prometheus (AMP) terdiri dari scraper yang menemukan dan mengumpulkan metrik dari sumber daya di kluster Amazon EKS. AMP mengelola scraper untuk Anda, menghilangkan kebutuhan untuk mengelola instans, agen, atau pencakar apa pun sendiri.

Anda dapat menggunakan kolektor terkelola AMP tanpa konfigurasi tambahan khusus untuk node hibrid. Namun titik akhir metrik untuk aplikasi Anda pada node hybrid harus dapat dijangkau dari VPC, termasuk rute dari VPC ke jaringan pod jarak jauh CIDRs dan port yang terbuka di firewall lokal Anda. Selain itu, kluster Anda harus memiliki [akses endpoint cluster pribadi](#).

Ikuti langkah-langkah dalam [Menggunakan kolektor AWS terkelola](#) di Amazon Managed Service for Prometheus User Guide.

AWS Distro untuk OpenTelemetry (ADOT)

Anda dapat menggunakan add-on AWS Distro for OpenTelemetry (ADOT) untuk mengumpulkan metrik, log, dan melacak data dari aplikasi Anda yang berjalan pada node hybrid. ADOT menggunakan [webhook](#) masuk untuk mengubah dan memvalidasi permintaan Sumber Daya Kustom Kolektor. Jika Anda menjalankan operator ADOT pada node hibrid, CIDR pod lokal Anda harus dapat dirutekan di jaringan lokal dan Anda harus mengonfigurasi kluster EKS dengan jaringan pod jarak jauh. Untuk informasi selengkapnya, lihat [Mengonfigurasi webhook untuk node hybrid](#).

Ikuti langkah-langkah dalam [Memulai dengan AWS Distro untuk OpenTelemetry menggunakan EKS Add-On](#) di AWS Distro untuk dokumentasi. OpenTelemetry

AWS Pengontrol Load Balancer

Anda dapat menggunakan [AWS Load Balancer Controller](#) dan Application Load Balancer (ALB) atau Network Load Balancer (NLB) dengan tipe target ip untuk beban kerja pada node hybrid yang terhubung dengan Direct Connect atau VPN. AWS Site-to-Site Target IP yang digunakan dengan ALB atau NLB harus dapat dirutekan dari. AWS [Pengontrol AWS Load Balancer juga menggunakan webhook](#). Jika Anda menjalankan operator AWS Load Balancer Controller pada node hybrid, CIDR pod lokal Anda harus dapat dirutekan di jaringan lokal dan Anda harus mengonfigurasi kluster EKS dengan jaringan pod jarak jauh. Untuk informasi selengkapnya, lihat [Mengonfigurasi webhook untuk node hybrid](#).

Untuk menginstal AWS Load Balancer Controller, ikuti langkah-langkah di [the section called “Instal dengan Helm”](#) atau [the section called “Instal dengan manifes”](#)

Untuk masuk dengan ALB, Anda harus menentukan anotasi di bawah ini. Lihat [the section called “Penyeimbangan beban aplikasi”](#) untuk instruksi.

```
alb.ingress.kubernetes.io/target-type: ip
```

Untuk load balancing dengan NLB, Anda harus menentukan anotasi di bawah ini. Lihat [the section called “Menyeimbangkan beban jaringan”](#) untuk instruksi.

```
service.beta.kubernetes.io/aws-load-balancer-type: "external"  
service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: "ip"
```

Agen Identitas EKS Pod

Note

Agar berhasil menerapkan add-on EKS Pod Identity Agent pada node hybrid yang menjalankan Bottlerocket, pastikan versi Bottlerocket Anda setidaknya v1.39.0. Agen Identitas Pod tidak didukung pada versi Bottlerocket sebelumnya di lingkungan node hybrid.

Agen Identitas Pod Amazon DaemonSet EKS asli bergantung pada ketersediaan EC2 IMDS pada node untuk mendapatkan kredensi yang diperlukan AWS . Karena IMDS tidak tersedia pada node hybrid, dimulai dengan versi 1.3.3-eksbuild.1, add-on Pod Identity Agent secara opsional menerapkan a yang memasang kredensil yang diperlukan. DaemonSet Node hybrid yang menjalankan Bottlerocket memerlukan metode yang berbeda untuk me-mount kredensialnya, dan mulai dari versi 1.3.7-eksbuild.2, add-on Pod Identity Agent secara opsional menyebarkan a yang secara khusus menargetkan node hybrid Bottlerocket. DaemonSet Bagian berikut menjelaskan proses untuk mengaktifkan opsional DaemonSets.

Ubuntu/RHEL/AL2023

1. Untuk menggunakan agen Pod Identity pada node hybrid Ubuntu/RHEL/AI 2023, atur `enableCredentialsFile: true` di bagian hybrid nodeadm konfigurasi seperti yang ditunjukkan di bawah ini:

```
apiVersion: node.eks.aws/v1alpha1  
kind: NodeConfig  
spec:  
  hybrid:
```

```
enableCredentialsFile: true # <-- Add this
```

Ini akan dikonfigurasi nodeadm untuk membuat file kredensial yang akan dikonfigurasi pada node di bawah/eks-hybrid/.aws/credentials, yang akan digunakan oleh eks-pod-identity-agent pod. File kredensial ini akan berisi AWS kredensial sementara yang akan di-refresh secara berkala.

2. Setelah Anda memperbarui nodeadm konfigurasi pada setiap node, jalankan `nodeadm init` perintah berikut dengan Anda `nodeConfig.yaml` untuk menggabungkan node hybrid Anda ke cluster Amazon EKS Anda. Jika node Anda telah bergabung dengan cluster sebelumnya, tetap jalankan `init` perintah lagi.

```
nodeadm init -c file://nodeConfig.yaml
```

3. Instal eks-pod-identity-agent dengan dukungan untuk node hybrid diaktifkan, dengan menggunakan AWS CLI atau. AWS Management Console
 - a. AWS CLI: Dari mesin yang Anda gunakan untuk mengelola cluster, jalankan perintah berikut untuk menginstal eks-pod-identity-agent dengan dukungan untuk node hybrid diaktifkan. Ganti `my-cluster` dengan nama klaster Anda.

```
aws eks create-addon \
  --cluster-name my-cluster \
  --addon-name eks-pod-identity-agent \
  --configuration-values '{"daemonsets":{"hybrid":{"create": true}}}'
```

- b. AWS Management Console: Jika Anda menginstal add-on Pod Identity Agent melalui AWS konsol, tambahkan berikut ini ke konfigurasi opsional untuk menyebarkan node hybrid DaemonSet yang menargetkan.

```
{"daemonsets":{"hybrid":{"create": true}}}
```

Bottlerocket

1. Untuk menggunakan agen Pod Identity pada node hybrid Bottlerocket, tambahkan `--enable-credentials-file=true` flag ke perintah yang digunakan untuk data pengguna container bootstrap Bottlerocket, seperti yang dijelaskan dalam [the section called “Hubungkan node hybrid dengan Bottlerocket”](#)
 - a. Jika Anda menggunakan penyedia kredensi SSM, perintah Anda akan terlihat seperti ini:

```
eks-hybrid-ssm-setup --activation-id=<activation-id> --activation-
code=<activation-code> --region=<region> --enable-credentials-file=true
```

- b. Jika Anda menggunakan penyedia kredensi IAM Roles Anywhere, perintah Anda akan terlihat seperti ini:

```
eks-hybrid-iam-ra-setup --certificate=<certificate> --key=<private-key> --enable-
credentials-file=true
```

Ini akan mengkonfigurasi skrip bootstrap untuk membuat file kredensial pada node di bawah `/var/eks-hybrid/.aws/credentials`, yang akan digunakan oleh `eks-pod-identity-agent` pod. File kredensial ini akan berisi AWS kredensial sementara yang akan di-refresh secara berkala.

2. Instal `eks-pod-identity-agent` dengan dukungan untuk node hybrid Bottlerocket diaktifkan, dengan menggunakan CLI atau AWS Management Console
 - a. AWS CLI: Dari mesin yang Anda gunakan untuk mengelola cluster, jalankan perintah berikut untuk menginstal `eks-pod-identity-agent` dengan dukungan untuk node hybrid Bottlerocket diaktifkan. Ganti `my-cluster` dengan nama kluster Anda.

```
aws eks create-addon \
  --cluster-name my-cluster \
  --addon-name eks-pod-identity-agent \
  --configuration-values '{"daemonsets":{"hybrid-bottlerocket":{"create":
true}}}'
```

- b. AWS Management Console: Jika Anda menginstal add-on Pod Identity Agent melalui AWS konsol, tambahkan berikut ini ke konfigurasi opsional untuk menyebarkan node hybrid Bottlerocket DaemonSet yang menargetkan Bottlerocket.

```
{"daemonsets":{"hybrid-bottlerocket":{"create": true}}}
```

Pengontrol snapshot CSI

Dimulai dengan `versiv8.1.0-eksbuild.2`, [add-on pengontrol snapshot CSI](#) menerapkan aturan anti-afinitas lunak untuk node hibrida, lebih memilih pengontrol deployment untuk berjalan EC2 di Wilayah yang sama dengan bidang kontrol Amazon AWS EKS. Menempatkan bersama deployment di AWS Wilayah yang sama dengan bidang kontrol Amazon EKS meningkatkan latensi.

Pengaya komunitas

Bagian berikut menjelaskan perbedaan antara menjalankan add-on komunitas yang kompatibel pada node hibrida dibandingkan dengan jenis komputasi Amazon EKS lainnya.

Server Metrik Kubernetes

Bidang kontrol perlu mencapai IP pod Metrics Server (atau IP node jika HostNetwork diaktifkan). Oleh karena itu, kecuali Anda menjalankan Metrics Server dalam mode HostNetwork, Anda harus mengonfigurasi jaringan pod jarak jauh saat membuat cluster Amazon EKS Anda, dan Anda harus membuat alamat IP pod Anda dapat dirutekan. Menerapkan Border Gateway Protocol (BGP) dengan CNI adalah salah satu cara umum untuk membuat alamat IP pod Anda dapat dirutekan.

manajer sertifikat

cert-manager menggunakan [webhooks](#). Jika Anda menjalankan cert-manager node hibrid, CIDR pod lokal Anda harus dapat dirutekan di jaringan lokal dan Anda harus mengonfigurasi kluster EKS dengan jaringan pod jarak jauh. Untuk informasi selengkapnya, lihat [Mengonfigurasi webhook untuk node hybrid](#).

Konfigurasikan webhook untuk node hybrid

Halaman ini merinci pertimbangan untuk menjalankan webhook dengan node hybrid. Webhook digunakan dalam aplikasi Kubernetes dan proyek open source, seperti Load Balancer AWS Controller dan CloudWatch Observability Agent, untuk melakukan kemampuan mutasi dan validasi saat runtime.

Jaringan pod yang dapat dirutekan

Jika Anda dapat membuat CIDR pod lokal dapat dirutekan di jaringan lokal, Anda dapat menjalankan webhook pada node hibrid. Ada beberapa teknik yang dapat Anda gunakan untuk membuat pod lokal CIDR dapat dirutekan di jaringan lokal Anda termasuk Border Gateway Protocol (BGP), rute statis, atau solusi perutean kustom lainnya. BGP adalah solusi yang direkomendasikan karena lebih skalabel dan lebih mudah dikelola daripada solusi alternatif yang memerlukan konfigurasi rute khusus atau manual. AWS mendukung kemampuan BGP Cilium dan Calico untuk pod iklan CIDRs, lihat [the section called “Konfigurasikan CNI”](#) dan untuk informasi lebih lanjut. [the section called “Pod jarak jauh yang dapat dirutekan CIDRs”](#)

Jaringan pod yang tidak dapat dirutekan

Jika Anda tidak dapat membuat CIDR pod lokal dapat dirutekan di jaringan lokal dan perlu menjalankan webhook, sebaiknya jalankan semua webhook di node cloud di cluster EKS yang sama dengan node hibrid Anda.

Pertimbangan untuk cluster mode campuran

Cluster mode campuran didefinisikan sebagai cluster EKS yang memiliki node hybrid dan node yang berjalan di AWS Cloud. Saat menjalankan cluster mode campuran, pertimbangkan rekomendasi berikut:

- Jalankan VPC CNI pada node di AWS Cloud dan Cilium atau Calico pada node hybrid. Cilium dan Calico tidak didukung AWS saat berjalan di node di Cloud. AWS
- Konfigurasi webhook untuk berjalan di node di AWS Cloud. Lihat [the section called “Konfigurasi webhook untuk add-on”](#) cara mengonfigurasi webhook untuk AWS dan add-on komunitas.
- Jika aplikasi Anda memerlukan pod yang berjalan di node di AWS Cloud untuk berkomunikasi langsung dengan pod yang berjalan pada node hybrid (“komunikasi timur-barat”), dan Anda menggunakan CNI VPC pada node di AWS Cloud, dan Cilium atau Calico pada node hybrid, maka CIDR pod lokal Anda harus dapat dirutekan di jaringan lokal Anda.
- Jalankan setidaknya satu replika CoreDNS pada node AWS di Cloud dan setidaknya satu replika CoreDNS pada node hybrid.
- Konfigurasi Distribusi Lalu Lintas Layanan untuk menjaga lalu lintas Layanan lokal ke zona asalnya. Untuk informasi selengkapnya tentang Distribusi Lalu Lintas Layanan, lihat [the section called “Konfigurasi Distribusi Lalu Lintas Layanan”](#).
- Jika Anda menggunakan AWS Application Load Balancers (ALB) atau Network Load Balancers (NLB) untuk lalu lintas beban kerja yang berjalan pada node hybrid, maka target IP yang digunakan dengan ALB atau NLB harus dapat dirutekan dari. AWS
- Add-on Metrics Server memerlukan konektivitas dari bidang kontrol EKS ke alamat IP pod Metrics Server. Jika Anda menjalankan add-on Metrics Server pada node hibrid, maka CIDR pod lokal Anda harus dapat dirutekan di jaringan lokal Anda.
- Untuk mengumpulkan metrik node hibrid menggunakan Amazon Managed Service untuk kolektor terkelola Prometheus (AMP), CIDR pod lokal Anda harus dapat dirutekan di jaringan lokal. Atau, Anda dapat menggunakan kolektor terkelola AMP untuk metrik dan sumber daya bidang kontrol EKS yang berjalan di AWS Cloud, dan add-on AWS Distro for OpenTelemetry (ADOT) untuk mengumpulkan metrik untuk node hibrid.

Konfigurasi cluster mode campuran

Untuk melihat webhook yang bermutasi dan memvalidasi yang berjalan di kluster, Anda dapat melihat jenis sumber daya Ekstensi di panel Resources konsol EKS untuk kluster Anda, atau Anda dapat menggunakan perintah berikut. EKS juga melaporkan metrik webhook di dasbor observabilitas cluster, lihat [the section called “Dasbor observabilitas”](#) untuk informasi lebih lanjut.

```
kubectl get mutatingwebhookconfigurations
```

```
kubectl get validatingwebhookconfigurations
```

Konfigurasi Distribusi Lalu Lintas Layanan

Saat menjalankan cluster mode campuran, kami menyarankan Anda menggunakan [Distribusi Lalu Lintas Layanan](#) untuk menjaga lalu lintas Layanan lokal ke zona asalnya. Service Traffic Distribution (tersedia untuk Kubernetes versi 1.31 dan yang lebih baru di EKS) adalah solusi yang direkomendasikan melalui [Topology Aware Routing](#) karena lebih mudah diprediksi. Dengan Distribusi Lalu Lintas Layanan, titik akhir yang sehat di zona tersebut akan menerima semua lalu lintas untuk zona itu. Dengan Topology Aware Routing, setiap layanan harus memenuhi beberapa kondisi di zona tersebut untuk menerapkan perutean kustom, jika tidak, ia merutekan lalu lintas secara merata ke semua titik akhir.

Jika Anda menggunakan Cilium sebagai CNI Anda, Anda harus menjalankan CNI dengan `enable-service-topology` set untuk mengaktifkan Distribusi Lalu Lintas Layanan. Anda dapat meneruskan konfigurasi ini dengan bendera Helm `install --set loadBalancer.serviceTopology=true` atau Anda dapat memperbarui instalasi yang ada dengan perintah Cilium CLI `cilium config set enable-service-topology true` Agen Cilium yang berjalan pada setiap node harus dimulai ulang setelah memperbarui konfigurasi untuk instalasi yang ada.

Contoh cara mengonfigurasi Distribusi Lalu Lintas Layanan untuk Layanan CoreDNS ditampilkan di bagian berikut, dan kami menyarankan Anda mengaktifkan hal yang sama untuk semua Layanan di kluster Anda untuk menghindari lalu lintas lintas lingkungan yang tidak diinginkan.

Konfigurasi replika CoreDNS

Saat menjalankan cluster mode campuran, sebaiknya Anda memiliki setidaknya satu replika CoreDNS pada node hybrid dan setidaknya satu replika CoreDNS pada node di Cloud. AWS

1. Tambahkan label zona topologi untuk setiap node hibrida Anda, misalnya.
`topology.kubernetes.io/zone: onprem` Atau, Anda dapat mengatur label pada nodeadm init fase dengan menentukan label dalam nodeadm konfigurasi Anda, lihat [the section called “Node Config untuk menyesuaikan kubelet \(Opsional\)”](#). Catatan, node yang berjalan di AWS Cloud secara otomatis mendapatkan label zona topologi yang diterapkan padanya yang sesuai dengan zona ketersediaan (AZ) node.

```
kubectl label node hybrid-node-name topology.kubernetes.io/zone=zone
```

2. Tambahkan podAntiAffinity ke penerapan CoreDNS dengan kunci zona topologi. Atau, Anda dapat mengonfigurasi penerapan CoreDNS selama instalasi dengan add-on EKS.

```
kubectl edit deployment coredns -n kube-system
```

```
spec:
  template:
    spec:
      affinity:
        ...
        podAntiAffinity:
          preferredDuringSchedulingIgnoredDuringExecution:
            - podAffinityTerm:
                labelSelector:
                  matchExpressions:
                    - key: k8s-app
                      operator: In
                      values:
                        - kube-dns
                topologyKey: kubernetes.io/hostname
              weight: 100
            - podAffinityTerm:
                labelSelector:
                  matchExpressions:
                    - key: k8s-app
                      operator: In
                      values:
                        - kube-dns
                topologyKey: topology.kubernetes.io/zone
              weight: 50
          ...
```

3. Tambahkan pengaturan ke konfigurasi kube-dns Layanan trafficDistribution: PreferClose untuk mengaktifkan Distribusi Lalu Lintas Layanan.

```
kubectl patch svc kube-dns -n kube-system --type=merge -p '{
  "spec": {
    "trafficDistribution": "PreferClose"
  }
}'
```

4. Anda dapat mengonfirmasi bahwa Distribusi Lalu Lintas Layanan diaktifkan dengan melihat irisan titik akhir untuk kube-dns Layanan. Irisan titik akhir Anda harus menunjukkan label zona topologi Anda, yang mengonfirmasi bahwa Distribusi Lalu Lintas Layanan diaktifkan. **hints** Jika Anda tidak melihat **hints** untuk setiap alamat titik akhir, maka Distribusi Lalu Lintas Layanan tidak diaktifkan.

```
kubectl get endpointslice -A | grep "kube-dns"
```

```
kubectl get endpointslice [replaceable]`kube-dns-` -n kube-system -o yaml
```

```
addressType: IPv4
apiVersion: discovery.k8s.io/v1
endpoints:
- addresses:
  - <your-hybrid-node-pod-ip>
  hints:
    forZones:
    - name: onprem
    nodeName: <your-hybrid-node-name>
    zone: onprem
- addresses:
  - <your-cloud-node-pod-ip>
  hints:
    forZones:
    - name: us-west-2a
    nodeName: <your-cloud-node-name>
    zone: us-west-2a
```

Konfigurasi webhook untuk add-on

Add-on berikut menggunakan webhook dan didukung untuk digunakan dengan node hybrid.

- AWS Pengontrol Load Balancer
- CloudWatch Agen Observabilitas
- AWS Distro untuk OpenTelemetry (ADOT)
- `cert-manager`

Lihat bagian berikut untuk mengonfigurasi webhook yang digunakan oleh add-on ini untuk berjalan di node di Cloud. AWS

AWS Pengontrol Load Balancer

Untuk menggunakan AWS Load Balancer Controller dalam pengaturan cluster mode campuran, Anda harus menjalankan controller pada node di AWS Cloud. Untuk melakukannya, tambahkan berikut ini ke konfigurasi nilai Helm Anda atau tentukan nilainya dengan menggunakan konfigurasi add-on EKS.

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
        - matchExpressions:
            - key: eks.amazonaws.com/compute-type
              operator: NotIn
              values:
                - hybrid
```

CloudWatch Agen Observabilitas

Add-on CloudWatch Observability Agent memiliki Operator Kubernetes yang menggunakan webhook. Untuk menjalankan operator pada node di AWS Cloud dalam pengaturan cluster mode campuran, edit konfigurasi operator CloudWatch Observability Agent. Anda tidak dapat mengonfigurasi afinitas operator selama penginstalan dengan add-on Helm dan EKS (lihat masalah [containers-roadmap #2431](#)).

```
kubectl edit -n amazon-cloudwatch deployment amazon-cloudwatch-observability-
controller-manager
```

```
spec:
  ...
  template:
    ...
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
              - matchExpressions:
                  - key: eks.amazonaws.com/compute-type
                    operator: NotIn
                    values:
                      - hybrid
```

AWS Distro untuk OpenTelemetry (ADOT)

Add-on AWS Distro for OpenTelemetry (ADOT) memiliki Operator Kubernetes yang menggunakan webhook. Untuk menjalankan operator pada node di AWS Cloud dalam pengaturan cluster mode campuran, tambahkan berikut ini ke konfigurasi nilai Helm Anda atau tentukan nilai dengan menggunakan konfigurasi add-on EKS.

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
        - matchExpressions:
            - key: eks.amazonaws.com/compute-type
              operator: NotIn
              values:
                - hybrid
```

Jika CIDR pod Anda tidak dapat dirutekan di jaringan lokal Anda, maka kolektor ADOT harus berjalan pada node hybrid untuk mengikis metrik dari node hibrid Anda dan beban kerja yang berjalan di dalamnya. Untuk melakukannya, edit Custom Resource Definition (CRD).

```
kubectl -n opentelemetry-operator-system edit opentelemetrycollectors.opentelemetry.io
adot-col-prom-metrics
```

```
spec:
  affinity:
```

```
nodeAffinity:
  requiredDuringSchedulingIgnoredDuringExecution:
    nodeSelectorTerms:
      - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: In
            values:
              - hybrid
```

Anda dapat mengonfigurasi kolektor ADOT untuk hanya mengikis metrik dari node hibrida dan sumber daya yang berjalan pada node hibrida dengan menambahkan yang berikut `relabel_configs` ke masing-masing `scrape_configs` dalam konfigurasi CRD kolektor ADOT.

```
relabel_configs:
  - action: keep
    regex: hybrid
    source_labels:
      - __meta_kubernetes_node_label_eks_aws_com_compute_type
```

Add-on ADOT memiliki persyaratan prasyarat `cert-manager` untuk menginstal sertifikat TLS yang digunakan oleh webhook operator ADOT. `cert-manager` juga menjalankan webhooks dan Anda dapat mengonfigurasinya untuk berjalan di node di AWS Cloud dengan konfigurasi nilai Helm berikut.

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
        - matchExpressions:
            - key: eks.amazonaws.com/compute-type
              operator: NotIn
              values:
                - hybrid
webhook:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
          - matchExpressions:
              - key: eks.amazonaws.com/compute-type
                operator: NotIn
                values:
                  - hybrid
```

```
cainjector:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
          - matchExpressions:
              - key: eks.amazonaws.com/compute-type
                operator: NotIn
                values:
                  - hybrid
startupapicheck:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
          - matchExpressions:
              - key: eks.amazonaws.com/compute-type
                operator: NotIn
                values:
                  - hybrid
```

cert-manager

cert-manager Add-on menjalankan webhooks dan Anda dapat mengonfigurasinya untuk berjalan di node di AWS Cloud dengan konfigurasi nilai Helm berikut.

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
        - matchExpressions:
            - key: eks.amazonaws.com/compute-type
              operator: NotIn
              values:
                - hybrid
webhook:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
          - matchExpressions:
              - key: eks.amazonaws.com/compute-type
                operator: NotIn
```

```
        values:
        - hybrid
cainjector:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
            - key: eks.amazonaws.com/compute-type
              operator: NotIn
              values:
              - hybrid
startupapicheck:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
            - key: eks.amazonaws.com/compute-type
              operator: NotIn
              values:
              - hybrid
```

Konfigurasi proxy untuk node hybrid

Jika Anda menggunakan server proxy di lingkungan lokal untuk lalu lintas yang meninggalkan pusat data atau lingkungan edge, Anda perlu mengonfigurasi node dan kluster secara terpisah untuk menggunakan server proxy Anda.

Kluster

Di cluster Anda, Anda perlu mengonfigurasi kube-proxy untuk menggunakan server proxy Anda. Anda harus mengonfigurasi kube-proxy setelah membuat cluster Amazon EKS Anda.

Simpul

Pada node Anda, Anda harus mengonfigurasi sistem operasi, containerd, kubelet, dan agen SSM Amazon untuk menggunakan server proxy Anda. Anda dapat membuat perubahan ini selama proses build untuk image sistem operasi Anda atau sebelum Anda menjalankan nodeadm init pada setiap node hybrid.

Konfigurasi tingkat simpul

Anda harus menerapkan konfigurasi berikut baik dalam gambar sistem operasi Anda atau sebelum berjalan `nodeadm init` pada setiap node hybrid.

containerd konfigurasi proxy

containerd adalah runtime manajemen kontainer default untuk Kubernetes. Jika Anda menggunakan proxy untuk akses internet, Anda harus mengkonfigurasi **containerd** sehingga dapat menarik gambar kontainer yang diperlukan oleh Kubernetes dan Amazon EKS.

Buat file pada setiap node hybrid yang disebut `http-proxy.conf` dalam `/etc/systemd/system/containerd.service.d` direktori dengan konten berikut. Ganti `proxy-domain` dan `port` dengan nilai-nilai untuk lingkungan Anda.

```
[Service]
Environment="HTTP_PROXY=http://proxy-domain:port"
Environment="HTTPS_PROXY=http://proxy-domain:port"
Environment="NO_PROXY=localhost"
```

containerd konfigurasi dari data pengguna

`containerd.service.d` Direktori harus dibuat untuk file ini. Anda perlu memuat ulang `systemd` untuk mengambil file konfigurasi tanpa reboot. Di AL2 023, layanan kemungkinan sudah berjalan ketika skrip Anda dijalankan, jadi Anda juga perlu memulai ulang.

```
mkdir -p /etc/systemd/system/containerd.service.d
echo '[Service]' > /etc/systemd/system/containerd.service.d/http-proxy.conf
echo 'Environment="HTTP_PROXY=http://proxy-domain:port"' >> /etc/systemd/system/
containerd.service.d/http-proxy.conf
echo 'Environment="HTTPS_PROXY=http://proxy-domain:port"' >> /etc/systemd/system/
containerd.service.d/http-proxy.conf
echo 'Environment="NO_PROXY=localhost"' >> /etc/systemd/system/containerd.service.d/
http-proxy.conf
systemctl daemon-reload
systemctl restart containerd
```

kubelet konfigurasi proxy

kubelet adalah agen node Kubernetes yang berjalan pada setiap node Kubernetes dan bertanggung jawab untuk mengelola node dan pod yang berjalan di atasnya. Jika Anda

menggunakan proxy di lingkungan lokal, Anda harus mengonfigurasinya kubelet agar dapat berkomunikasi dengan titik akhir publik atau pribadi kluster Amazon EKS Anda.

Buat file pada setiap node hybrid yang disebut `http-proxy.conf` dalam `/etc/systemd/system/kubelet.service.d/` direktori dengan konten berikut. Ganti `proxy-domain` dan `port` dengan nilai-nilai untuk lingkungan Anda.

```
[Service]
Environment="HTTP_PROXY=http://proxy-domain:port"
Environment="HTTPS_PROXY=http://proxy-domain:port"
Environment="NO_PROXY=localhost"
```

kubelet konfigurasi dari data pengguna

`kubelet.service.d` Direktori harus dibuat untuk file ini. Anda perlu memuat ulang `systemd` untuk mengambil file konfigurasi tanpa reboot. Di AL2 023, layanan kemungkinan sudah berjalan ketika skrip Anda dijalankan, jadi Anda juga perlu memulai ulang.

```
mkdir -p /etc/systemd/system/kubelet.service.d
echo '[Service]' > /etc/systemd/system/kubelet.service.d/http-proxy.conf
echo 'Environment="HTTP_PROXY=http://proxy-domain:port"' >> /etc/systemd/system/
kubelet.service.d/http-proxy.conf
echo 'Environment="HTTPS_PROXY=http://proxy-domain:port"' >> /etc/systemd/system/
kubelet.service.d/http-proxy.conf
echo 'Environment="NO_PROXY=localhost"' >> /etc/systemd/system/kubelet.service.d/http-
proxy.conf
systemctl daemon-reload
systemctl restart kubelet
```

ssm konfigurasi proxy

`ssm` adalah salah satu penyedia kredensi yang dapat digunakan untuk menginisialisasi node hybrid. `ssm` bertanggung jawab untuk mengautentikasi dengan AWS dan menghasilkan kredensi sementara yang digunakan oleh `kubelet`. Jika Anda menggunakan proxy di lingkungan lokal dan menggunakan `ssm` sebagai penyedia kredensi di node, Anda harus mengonfigurasinya `ssm` agar dapat berkomunikasi dengan titik akhir layanan Amazon SSM.

Buat file pada setiap node hybrid yang disebut `http-proxy.conf` di jalur di bawah ini tergantung pada sistem operasi

- Ubuntu - `/etc/systemd/system/snap.amazon-ssm-agent.amazon-ssm-agent.service.d/http-proxy.conf`
- Amazon Linux 2023 dan Red Hat Enterprise Linux - `/etc/systemd/system/amazon-ssm-agent.service.d/http-proxy.conf`

Isi file dengan konten berikut. Ganti proxy-domain dan port dengan nilai-nilai untuk lingkungan Anda.

```
[Service]
Environment="HTTP_PROXY=http://proxy-domain:port"
Environment="HTTPS_PROXY=http://proxy-domain:port"
Environment="NO_PROXY=localhost"
```

ssmkonfigurasi dari data pengguna

Direktori file layanan ssm systemd harus dibuat untuk file ini. Jalur direktori tergantung pada sistem operasi yang digunakan pada node.

- Ubuntu - `/etc/systemd/system/snap.amazon-ssm-agent.amazon-ssm-agent.service.d`
- Amazon Linux 2023 dan Red Hat Enterprise Linux - `/etc/systemd/system/amazon-ssm-agent.service.d`

Ganti nama layanan systemd dalam perintah restart di bawah ini tergantung pada sistem operasi yang digunakan pada node

- Ubuntu - `snap.amazon-ssm-agent.amazon-ssm-agent`
- Amazon Linux 2023 dan Red Hat Enterprise Linux - `amazon-ssm-agent`

```
mkdir -p systemd-service-file-directory
echo '[Service]' > [replaceable]#systemd-service-file-directory/http-proxy.conf
echo 'Environment="HTTP_PROXY=http://[replaceable]#proxy-domain:port"' >> systemd-service-file-directory/http-proxy.conf
echo 'Environment="HTTPS_PROXY=http://[replaceable]#proxy-domain:port"' >> [replaceable]#systemd-service-file-directory/http-proxy.conf
echo 'Environment="NO_PROXY=localhost"' >> [replaceable]#systemd-service-file-directory/http-proxy.conf
```

```
systemctl daemon-reload
systemctl restart [.replaceable]#systemd-service-name
```

Konfigurasi proxy sistem operasi

Jika Anda menggunakan proxy untuk akses internet, Anda harus mengkonfigurasi sistem operasi Anda untuk dapat menarik dependensi node hybrid dari manajer paket sistem operasi Anda.

Ubuntu

1. Konfigurasi snap untuk menggunakan proxy Anda dengan perintah berikut:

```
sudo snap set system proxy.https=http://proxy-domain:port
sudo snap set system proxy.http=http://proxy-domain:port
```

2. Untuk mengaktifkan proxyapt, buat file yang disebut `apt.conf` dalam `/etc/apt/` direktori. Ganti `proxy-domain` dan port dengan nilai untuk lingkungan Anda.

```
Acquire::http::Proxy "http://proxy-domain:port";
Acquire::https::Proxy "http://proxy-domain:port";
```

Amazon Linux 2023

1. `dnf` Konfigurasi untuk menggunakan proxy Anda. Buat file `/etc/dnf/dnf.conf` dengan `proxy-domain` dan nilai port untuk lingkungan Anda.

```
proxy=http://proxy-domain:port
```

Perusahaan Topi Merah Linux

1. `yum` Konfigurasi untuk menggunakan proxy Anda. Buat file `/etc/yum.conf` dengan `proxy-domain` dan nilai port untuk lingkungan Anda.

```
proxy=http://proxy-domain:port
```

Konfigurasi proxy Peran IAM Di Mana Saja

Layanan penyedia kredensi IAM Roles Anywhere bertanggung jawab untuk menyegarkan kredensial saat menggunakan Peran IAM Di Mana Saja dengan tanda (lihat). [enableCredentialsFile the section called “Agen Identitas EKS Pod”](#) Jika Anda menggunakan proxy di lingkungan lokal, Anda harus mengonfigurasi layanan agar dapat berkomunikasi dengan titik akhir IAM Roles Anywhere.

Buat file yang disebut `http-proxy.conf` dalam `/etc/systemd/system/aws_signing_helper_update.service.d/` direktori dengan konten berikut. Ganti `proxy-domain` dan `port` dengan nilai-nilai untuk lingkungan Anda.

```
[Service]
Environment="HTTP_PROXY=http://proxy-domain:port"
Environment="HTTPS_PROXY=http://proxy-domain:port"
Environment="NO_PROXY=localhost"
```

Konfigurasi luas cluster

Konfigurasi di bagian ini harus diterapkan setelah Anda membuat klaster Amazon EKS dan sebelum berjalan `nodeadm init` pada setiap node hybrid.

konfigurasi proxy kube-proxy

Amazon EKS secara otomatis menginstal `kube-proxy` pada setiap node hybrid sebagai `DaemonSet` saat node hybrid Anda bergabung dengan cluster. `kube-proxy` memungkinkan perutean di seluruh layanan yang didukung oleh pod di kluster Amazon EKS. Untuk mengonfigurasi setiap host, `kube-proxy` memerlukan resolusi DNS untuk titik akhir klaster Amazon EKS Anda.

1. Edit `kube-proxy` `DaemonSet` dengan perintah berikut

```
kubectl -n kube-system edit ds kube-proxy
```

Ini akan membuka `kube-proxy` `DaemonSet` definisi pada editor Anda yang dikonfigurasi.

2. Tambahkan variabel lingkungan untuk `HTTP_PROXY` dan `HTTPS_PROXY`. Perhatikan variabel `NODE_NAME` lingkungan seharusnya sudah ada dalam konfigurasi Anda. Ganti `proxy-domain` dan `port` dengan nilai untuk lingkungan Anda.

```
containers:
  - command:
    - kube-proxy
```

```
- --v=2
- --config=/var/lib/kube-proxy-config/config - --hostname-override=$(NODE_NAME)
env:
- name: HTTP_PROXY
  value: http://proxy-domain:port
- name: HTTPS_PROXY
  value: http://proxy-domain:port
- name: NODE_NAME
  valueFrom:
    fieldRef:
      apiVersion: v1
      fieldPath: spec.nodeName
```

Konsep untuk node hibrida

Dengan Amazon EKS Hybrid Nodes, Anda menggabungkan mesin fisik atau virtual yang berjalan di lingkungan lokal atau edge ke kluster Amazon EKS yang berjalan di AWS Cloud. Pendekatan ini membawa banyak manfaat, tetapi juga memperkenalkan konsep dan arsitektur jaringan baru bagi mereka yang akrab dengan menjalankan kluster Kubernetes dalam satu lingkungan jaringan.

Bagian berikut menyelam jauh ke dalam Kubernetes dan konsep jaringan untuk EKS Hybrid Nodes dan merinci bagaimana lalu lintas mengalir melalui arsitektur hybrid. Bagian ini mengharuskan Anda memahami pengetahuan dasar jaringan Kubernetes, seperti konsep pod, node, layanan, bidang kontrol Kubernetes, kubelet, dan kube-proxy.

Kami merekomendasikan membaca halaman-halaman ini secara berurutan [the section called “Konsep jaringan”](#), dimulai dengan [the section called “Konsep Kubernetes”](#), lalu, dan akhirnya [the section called “Arus lalu lintas”](#).

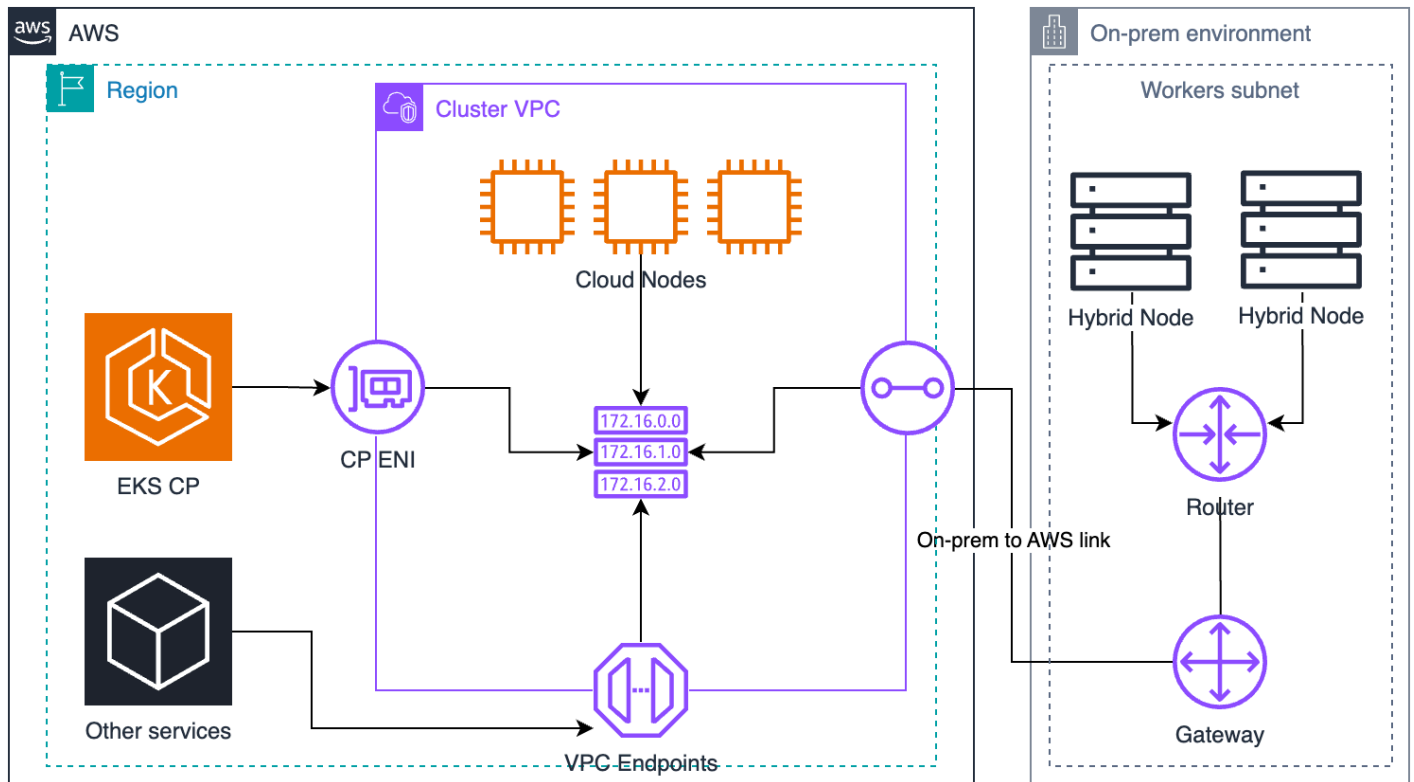
Topik

- [Konsep jaringan untuk node hibrida](#)
- [Konsep Kubernetes untuk node hibrida](#)
- [Arus lalu lintas jaringan untuk node hibrida](#)

Konsep jaringan untuk node hibrida

Bagian ini merinci konsep jaringan inti dan kendala yang harus Anda pertimbangkan saat merancang topologi jaringan Anda untuk EKS Hybrid Nodes.

Konsep jaringan untuk EKS Hybrid Nodes



VPC sebagai hub jaringan

Semua lalu lintas yang melintasi rute batas cloud melalui VPC Anda. Ini termasuk lalu lintas antara bidang kontrol EKS atau pod yang berjalan AWS ke node hibrida atau pod yang berjalan di atasnya. Anda dapat menganggap VPC cluster Anda sebagai hub jaringan antara node hybrid Anda dan cluster lainnya. Arsitektur ini memberi Anda kontrol penuh atas lalu lintas dan peruteannya tetapi juga menjadikannya tanggung jawab Anda untuk mengonfigurasi rute, grup keamanan, dan firewall dengan benar untuk VPC.

Pesawat kontrol EKS ke VPC

Bidang kontrol EKS memasang Elastic Network Interfaces (ENIs) ke VPC Anda. Ini ENIs menangani lalu lintas ke dan dari server API EKS. Anda mengontrol penempatan bidang kontrol EKS ENIs saat mengonfigurasi kluster, saat EKS ENIs menempel pada subnet yang Anda lewati selama pembuatan kluster.

EKS mengaitkan Grup Keamanan dengan EKS ENIs yang melekat pada subnet Anda. Kelompok keamanan ini memungkinkan lalu lintas ke dan dari pesawat kontrol EKS melalui ENIs. Ini penting

untuk EKS Hybrid Nodes karena Anda harus mengizinkan lalu lintas dari node hybrid dan pod yang berjalan di atasnya ke bidang kontrol EKS ENIs.

Jaringan Node Jarak Jauh

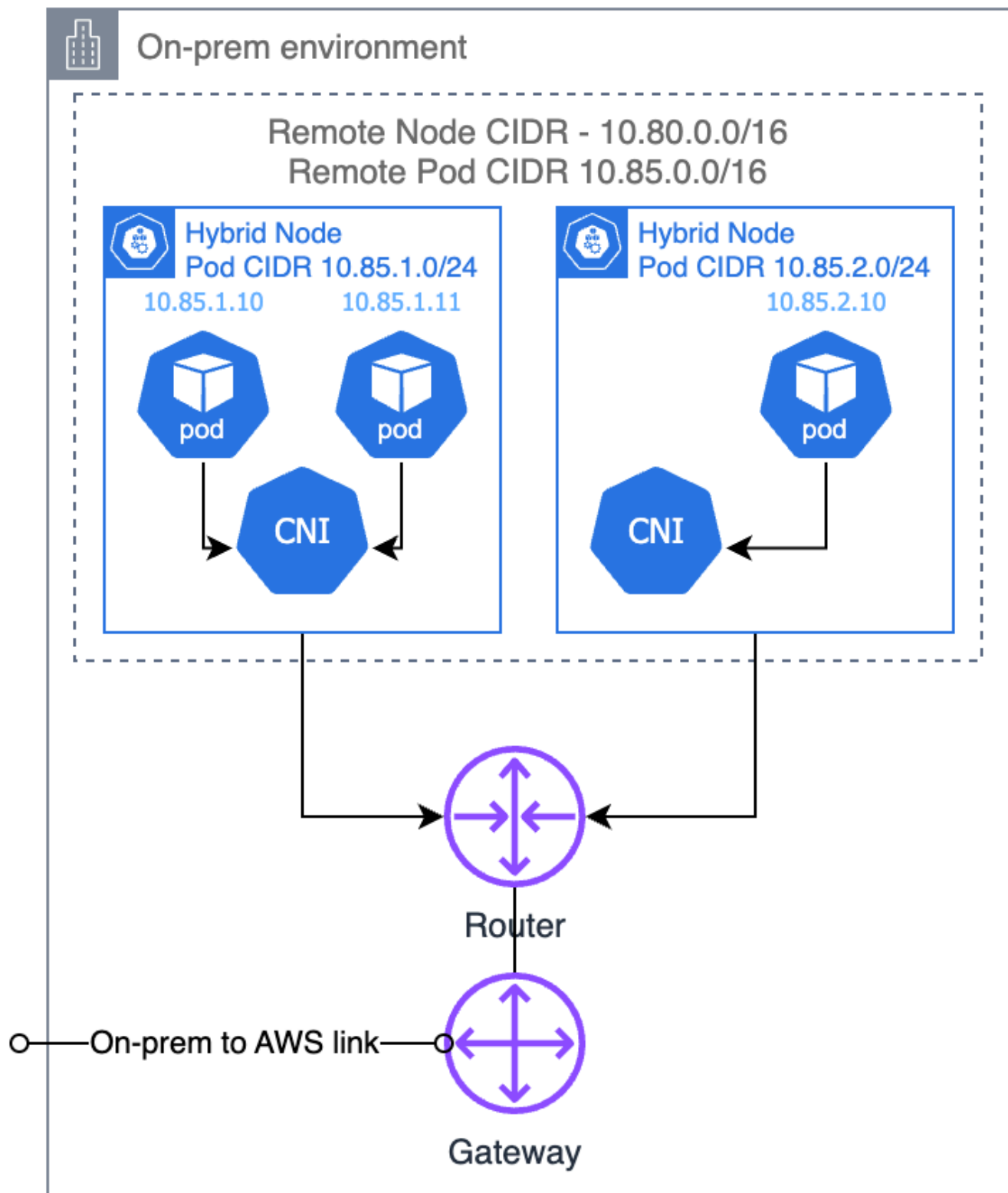
Jaringan node jarak jauh, khususnya node jarak jauh CIDRs, adalah rentang yang IPs ditugaskan ke mesin yang Anda gunakan sebagai node hibrida. Saat Anda menyediakan node hibrid, node tersebut berada di pusat data lokal atau lokasi edge Anda, yang merupakan domain jaringan yang berbeda dari bidang kontrol EKS dan VPC. Setiap node hybrid memiliki alamat IP, atau alamat, dari node jarak jauh CIDR yang berbeda dari subnet di VPC Anda.

Anda mengonfigurasi cluster EKS dengan node jarak jauh ini CIDRs sehingga EKS tahu untuk merutekan semua lalu lintas yang ditujukan untuk node hybrid IPs melalui VPC cluster Anda, seperti permintaan ke kubelet API.

Jaringan Pod Jarak Jauh

Jaringan pod jarak jauh adalah rentang yang IPs ditetapkan ke pod yang berjalan pada node hybrid. Umumnya, Anda mengonfigurasi CNI Anda dengan rentang ini dan fungsionalitas IP Address Management (IPAM) dari CNI menangani penetapan sepotong rentang ini ke setiap node hybrid. Saat Anda membuat pod, CNI memberikan IP ke pod dari irisan yang dialokasikan ke node tempat pod telah dijadwalkan.

Anda mengonfigurasi cluster EKS dengan pod jarak jauh ini CIDRs sehingga control plane EKS tahu untuk merutekan semua lalu lintas yang ditujukan untuk pod yang berjalan pada node hybrid melalui VPC klaster Anda, seperti komunikasi dengan webhook.



Lokal ke VPC

Jaringan lokal yang Anda gunakan untuk node hibrid harus merutekan ke VPC yang Anda gunakan untuk kluster EKS Anda. Ada beberapa [opsi konektivitas Network-to-Amazon VPC](#) yang tersedia untuk menghubungkan jaringan lokal Anda ke VPC. Anda juga dapat menggunakan solusi VPN Anda sendiri.

Penting bagi Anda untuk mengonfigurasi perutean dengan benar di sisi AWS Cloud di VPC dan di jaringan lokal Anda, sehingga kedua jaringan merutekan lalu lintas yang tepat melalui koneksi untuk kedua jaringan tersebut.

Di VPC, semua lalu lintas yang menuju ke node jarak jauh dan jaringan pod jarak jauh harus merutekan melalui koneksi ke jaringan lokal Anda (disebut sebagai “gateway”). Jika beberapa subnet Anda memiliki tabel rute yang berbeda, Anda harus mengonfigurasi setiap tabel rute dengan rute untuk node hibrida dan pod yang berjalan di atasnya. Hal ini berlaku untuk subnet di mana bidang kontrol EKS ENIs dilampirkan, dan subnet yang berisi EC2 node atau pod yang harus berkomunikasi dengan node hibrida.

Di jaringan lokal Anda, Anda harus mengonfigurasi jaringan Anda untuk mengizinkan lalu lintas ke dan dari VPC kluster EKS Anda dan layanan AWS lain yang diperlukan untuk node hibrid. Lalu lintas untuk kluster EKS melintasi gateway di kedua arah.

Kendala jaringan

Jaringan yang sepenuhnya dirutekan

Kendala utama adalah bahwa bidang kontrol EKS dan semua node, cloud atau node hybrid, perlu membentuk jaringan yang sepenuhnya dirutekan. Ini berarti bahwa semua node harus dapat menjangkau satu sama lain pada lapisan tiga, dengan alamat IP.

Bidang kontrol EKS dan node awan sudah dapat dijangkau satu sama lain karena mereka berada dalam jaringan datar (VPC). Node hybrid, bagaimanapun, berada dalam domain jaringan yang berbeda. Inilah sebabnya mengapa Anda perlu mengonfigurasi perutean tambahan di VPC dan di jaringan lokal Anda untuk merutekan lalu lintas antara node hibrida dan cluster lainnya. Jika node hybrid dapat dijangkau satu sama lain dan dari VPC, node hybrid Anda dapat berada dalam satu jaringan datar tunggal atau di beberapa jaringan tersegmentasi.

Pod jarak jauh yang dapat dirutekan CIDRs

Agar control plane EKS dapat berkomunikasi dengan pod yang berjalan pada node hybrid (misalnya, webhook atau Metrics Server) atau untuk pod yang berjalan di cloud node untuk berkomunikasi dengan pod yang berjalan pada node hybrid (beban kerja komunikasi timur-barat), pod jarak jauh

CIDR Anda harus dapat dirutekan dari VPC. Ini berarti bahwa VPC harus dapat merutekan lalu lintas ke pod CIDRs melalui gateway ke jaringan lokal Anda dan bahwa jaringan lokal Anda harus dapat merutekan lalu lintas pod ke node yang tepat.

Penting untuk dicatat perbedaan antara persyaratan perutean pod di VPC dan lokal. VPC hanya perlu tahu bahwa setiap lalu lintas yang menuju pod jarak jauh harus melalui gateway. Jika Anda hanya memiliki satu pod jarak jauh CIDR, Anda hanya perlu satu rute.

Persyaratan ini berlaku untuk semua hop di jaringan lokal Anda hingga router lokal di subnet yang sama dengan node hybrid Anda. Ini adalah satu-satunya router yang perlu mengetahui potongan CIDR pod yang ditetapkan untuk setiap node, memastikan bahwa lalu lintas untuk pod tertentu dikirim ke node tempat pod telah dijadwalkan.

Anda dapat memilih untuk menyebarkan rute ini untuk pod lokal CIDRs dari router lokal lokal Anda ke tabel rute VPC, tetapi itu tidak diperlukan. Jika pod lokal Anda sering CIDRs berubah dan tabel rute VPC Anda perlu diperbarui untuk mencerminkan pod yang berubah, sebaiknya Anda menyebarkan CIDRs pod lokal CIDRs ke tabel rute VPC, tetapi hal ini jarang terjadi.

Perhatikan, kendala untuk membuat pod lokal Anda dapat CIDRs dirutekan adalah opsional. Jika Anda tidak perlu menjalankan webhook pada node hybrid atau pod di cloud node berbicara dengan pod pada node hybrid, Anda tidak perlu mengonfigurasi routing untuk pod CIDRs di jaringan lokal Anda.

Mengapa pod lokal CIDRs harus dapat dirutekan dengan node hybrid?

Saat menggunakan EKS dengan VPC CNI untuk node cloud Anda, VPC CNI menugaskan langsung IPs dari VPC ke pod. Ini berarti tidak ada kebutuhan untuk routing khusus, karena kedua pod cloud dan bidang kontrol EKS dapat mencapai Pod secara IPs langsung.

Saat menjalankan on-premise (dan dengan yang lain CNIs di cloud), pod biasanya berjalan di jaringan overlay yang terisolasi dan CNI menangani pengiriman lalu lintas antar pod. Ini biasanya dilakukan melalui enkapsulasi: CNI mengubah pod-to-pod lalu lintas menjadi lalu node-to-node lintas, mengurus enkapsulasi dan de-enkapsulasi di kedua ujungnya. Dengan cara ini, tidak perlu konfigurasi tambahan pada node dan pada router.

Jaringan dengan node hibrida unik karena menyajikan kombinasi kedua topologi - bidang kontrol EKS dan node cloud (dengan VPC CNI) mengharuskan jaringan datar termasuk node dan pod, sedangkan pod yang berjalan pada node hibrida berada dalam jaringan overlay dengan menggunakan VXLAN untuk enkapsulasi (secara default di Cilium). Pod yang berjalan pada node hybrid dapat mencapai control plane EKS dan pod yang berjalan di cloud node dengan asumsi

jaringan lokal dapat merutekan ke VPC. Namun, tanpa merutekan pod CIDRs di jaringan lokal, lalu lintas apa pun yang kembali ke IP pod lokal akan dihapus pada akhirnya jika jaringan tidak tahu cara menjangkau jaringan overlay dan merutekan ke node yang benar.

Konsep Kubernetes untuk node hibrida

Halaman ini merinci konsep-konsep utama Kubernetes yang mendukung arsitektur sistem EKS Hybrid Nodes.

Pesawat kontrol EKS di VPC

IPs Bidang kontrol EKS ENIs disimpan dalam kubernetes Endpoints objek di default namespace. Ketika EKS membuat yang baru ENIs atau menghapus yang lebih lama, EKS memperbarui objek ini sehingga daftar IPs selalu up-to-date.

Anda dapat menggunakan titik akhir ini melalui kubernetes Layanan, juga di default namespace. Layanan ini, dari ClusterIP jenis, selalu diberi IP pertama dari CIDR layanan cluster. Misalnya, untuk layanan CIDR172.16.0.0/16, IP layanan akan172.16.0.1.

Secara umum, ini adalah bagaimana Pod (terlepas dari apakah berjalan di cloud atau node hybrid) mengakses server API EKS Kubernetes. Pod menggunakan IP layanan sebagai IP tujuan, yang diterjemahkan ke IPs salah satu bidang kontrol EKS yang sebenarnya ENIs. Pengecualian utama adalah kube-proxy, karena mengatur terjemahan.

Titik akhir server API EKS

IP kubernetes layanan bukan satu-satunya cara untuk mengakses server EKS API. EKS juga membuat nama DNS Route53 saat Anda membuat cluster Anda. Ini adalah endpoint bidang cluster EKS Anda saat memanggil aksi EKS DescribeCluster API.

```
{
  "cluster": {
    "endpoint": "https://xxxxxxxxxxxxxxxxxxxxxxxxxxxxx.gr7.us-
west-2.eks.amazonaws.com",
    "name": "my-cluster",
    "status": "ACTIVE"
  }
}
```

Dalam akses endpoint publik atau kluster akses endpoint publik dan pribadi, node hybrid Anda akan menyelesaikan nama DNS ini ke IP publik secara default, yang dapat dirutekan melalui internet.

Dalam kluster akses titik akhir pribadi, nama DNS diselesaikan menjadi pribadi IPs dari bidang kontrol EKS. ENIs

Ini adalah bagaimana kubelet dan kube-proxy mengakses server API Kubernetes. Jika Anda ingin semua lalu lintas kluster Kubernetes mengalir melalui VPC, Anda perlu mengonfigurasi kluster Anda dalam mode akses pribadi atau memodifikasi server DNS lokal Anda untuk menyelesaikan titik akhir kluster EKS ke privat bidang kontrol EKS. IPs ENIs

Titik akhir **kubelet**

Ini kubelet mengekspos beberapa titik akhir REST, memungkinkan bagian lain dari sistem untuk berinteraksi dengan dan mengumpulkan informasi dari setiap node. Di sebagian besar cluster, sebagian besar lalu lintas ke kubelet server berasal dari bidang kontrol, tetapi agen pemantauan tertentu mungkin juga berinteraksi dengannya.

Melalui antarmuka ini, kubelet menangani berbagai permintaan: mengambil log (`kubectl logs`), mengeksekusi perintah di dalam container (`kubectl exec`), dan port-forwarding traffic (`kubectl port-forward`). Setiap permintaan ini berinteraksi dengan runtime kontainer yang mendasarinya melalui kubelet, tampak mulus bagi administrator dan pengembang kluster.

Konsumen yang paling umum dari API ini adalah Kubernetes API server. Saat Anda menggunakan salah satu `kubectl` perintah yang disebutkan sebelumnya, `kubectl` buat permintaan API ke server API, yang kemudian memanggil kubelet API node tempat pod berjalan. Ini adalah alasan utama mengapa IP node harus dapat dijangkau dari bidang kontrol EKS dan mengapa, bahkan jika pod Anda berjalan, Anda tidak akan dapat mengakses log mereka atau `exec` jika rute node salah dikonfigurasi.

Node IPs

Ketika bidang kontrol EKS berkomunikasi dengan node, ia menggunakan salah satu alamat yang dilaporkan dalam status Node objek (`status.addresses`).

Dengan node cloud EKS, kubelet biasanya melaporkan IP pribadi EC2 instance sebagai `InternalIP` selama pendaftaran node. IP ini kemudian divalidasi oleh Cloud Controller Manager (CCM) memastikan itu milik instance. EC2 Selain itu, CCM biasanya menambahkan nama publik IPs (`asExternalIP`) dan DNS (`InternalDNS` dan `ExternalDNS`) instance ke status node.

Namun, tidak ada CCM untuk node hybrid. Ketika Anda mendaftarkan node hybrid dengan EKS Hybrid Nodes CLI (`nodeadm`), ia mengkonfigurasi kubelet untuk melaporkan IP mesin Anda secara langsung dalam status node, tanpa CCM.

```
apiVersion: v1
kind: Node
metadata:
  name: my-node-1
spec:
  providerID: eks-hybrid:///us-west-2/my-cluster/my-node-1
status:
  addresses:
    - address: 10.1.1.236
      type: InternalIP
    - address: my-node-1
      type: Hostname
```

Jika mesin Anda memiliki beberapa IPs, kubelet akan memilih salah satu dari mereka mengikuti logikanya sendiri. Anda dapat mengontrol IP yang dipilih dengan `--node-ip` bendera, yang dapat Anda berikan dalam `nodeadm spec.kubelet.flags` konfigurasi. Hanya IP yang dilaporkan dalam Node objek yang membutuhkan rute dari VPC. Mesin Anda dapat memiliki IPs yang lain yang tidak dapat dijangkau dari cloud.

kube-proxy

kube-proxy bertanggung jawab untuk mengimplementasikan abstraksi Layanan pada lapisan jaringan setiap node. Ini bertindak sebagai proxy jaringan dan penyeimbang beban untuk lalu lintas yang ditujukan ke Layanan Kubernetes. Dengan terus memantau server API Kubernetes untuk perubahan yang terkait dengan Layanan dan Titik Akhir, memperbarui aturan jaringan host yang mendasarinya kube-proxy secara dinamis untuk memastikan lalu lintas diarahkan dengan benar.

Dalam iptables mode, kube-proxy program beberapa netfilter rantai untuk menangani lalu lintas layanan. Aturan membentuk hierarki berikut:

1. Rantai KUBE-SERVICES: Titik masuk untuk semua lalu lintas layanan. Ini memiliki aturan yang cocok dengan setiap layanan ClusterIP dan port.
2. KUBE-SVC-XXX rantai: Rantai khusus layanan memiliki aturan penyeimbangan beban untuk setiap layanan.
3. KUBE-SEP-XXX rantai: Rantai khusus titik akhir memiliki aturan yang sebenarnya. DNAT

Mari kita periksa apa yang terjadi untuk sebuah layanan `test-server` di default namespace: *
Service ClusterIP: * Service port: 172.16.31.14 * Backing pods: 80,, dan 10.2.0.110 10.2.1.39 10.2.2.254

Saat kami memeriksa iptables aturan (menggunakan `iptables-save | grep -A10 KUBE-SERVICES`):

1. Dalam rantai KUBE-SERVICES, kami menemukan aturan yang cocok dengan layanan:

```
-A KUBE-SERVICES -d 172.16.31.14/32 -p tcp -m comment --comment "default/test-server cluster IP" -m tcp --dport 80 -j KUBE-SVC-XYZABC123456
```

- Aturan ini cocok dengan paket yang ditujukan untuk 172.16.31. 14:80
- Komentar menunjukkan untuk apa aturan ini: `default/test-server cluster IP`
- Paket yang cocok melompat ke rantai KUBE-SVC-XYZABC123456

2. XYZABC123456 Rantai KUBE-SVC memiliki aturan load balancing berbasis probabilitas:

```
-A KUBE-SVC-XYZABC123456 -m statistic --mode random --probability 0.33333333349 -j KUBE-SEP-POD1XYZABC
-A KUBE-SVC-XYZABC123456 -m statistic --mode random --probability 0.50000000000 -j KUBE-SEP-POD2XYZABC
-A KUBE-SVC-XYZABC123456 -j KUBE-SEP-POD3XYZABC
```

- Aturan pertama: 33,3% peluang untuk melompat KUBE-SEP-POD1XYZABC
- Aturan kedua: 50% kemungkinan lalu lintas yang tersisa (33,3% dari total) untuk melompat KUBE-SEP-POD2XYZABC
- Aturan terakhir: Semua lalu lintas yang tersisa (33,3% dari total) melompat ke KUBE-SEP-POD3XYZABC

3. Rantai KUBE-SEP-XXX individu melakukan DNAT (Destination NAT):

```
-A KUBE-SEP-POD1XYZABC -p tcp -m tcp -j DNAT --to-destination 10.2.0.110:80
-A KUBE-SEP-POD2XYZABC -p tcp -m tcp -j DNAT --to-destination 10.2.1.39:80
-A KUBE-SEP-POD3XYZABC -p tcp -m tcp -j DNAT --to-destination 10.2.2.254:80
```

- Aturan DNAT ini menulis ulang IP tujuan dan port untuk mengarahkan lalu lintas ke pod tertentu.
- Setiap aturan menangani sekitar 33,3% dari lalu lintas, memberikan keseimbangan beban yang merata antara 10.2.0.110, dan 10.2.1.39 10.2.2.254

Struktur rantai multi-level ini memungkinkan kube-proxy untuk secara efisien menerapkan penyeimbangan beban layanan dan pengalihan melalui manipulasi paket tingkat kernel, tanpa memerlukan proses proxy di jalur data.

Dampak pada operasi Kubernetes

Kerusakan kube-proxy pada node mencegah node tersebut merutekan lalu lintas Layanan dengan benar, menyebabkan batas waktu atau koneksi gagal untuk pod yang mengandalkan Layanan cluster. Ini bisa sangat mengganggu ketika node pertama kali terdaftar. CNI perlu berbicara dengan server API Kubernetes untuk mendapatkan informasi, seperti pod CIDR node, sebelum dapat mengkonfigurasi jaringan pod apa pun. Untuk melakukan itu, ia menggunakan IP kubernetes Layanan. Namun, jika kube-proxy belum dapat memulai atau gagal menetapkan iptables aturan yang tepat, permintaan yang masuk ke IP kubernetes layanan tidak diterjemahkan ke bidang kontrol EKS yang sebenarnya IPs ENIs. Akibatnya, CNI akan memasuki crash loop dan tidak ada pod yang dapat berjalan dengan baik.

Kita tahu pod menggunakan IP kubernetes layanan untuk berkomunikasi dengan server API Kubernetes, tetapi kube-proxy perlu terlebih dahulu menetapkan iptables aturan untuk membuatnya berfungsi.

Bagaimana cara kube-proxy berkomunikasi dengan server API?

kube-proxy Harus dikonfigurasi untuk menggunakan server API Kubernetes yang sebenarnya IP/s atau nama DNS yang menyelesaikannya. Dalam kasus EKS, EKS mengonfigurasi default kube-proxy untuk menunjuk ke nama DNS Route53 yang dibuat EKS saat Anda membuat cluster. Anda dapat melihat nilai ini di kube-proxy ConfigMap dalam kube-system namespace. Isi dari ini ConfigMap adalah a kubeconfig yang disuntikkan ke dalam kube-proxy pod, jadi cari `clusters0.cluster.server` bidangnya. Nilai ini akan cocok dengan endpoint bidang kluster EKS Anda (saat memanggil EKS DescribeCluster API).

```
apiVersion: v1
data:
  kubeconfig: |-
    kind: Config
    apiVersion: v1
    clusters:
    - cluster:
        certificate-authority: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
        server: https://xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.gr7.us-
west-2.eks.amazonaws.com
        name: default
    contexts:
    - context:
        cluster: default
```

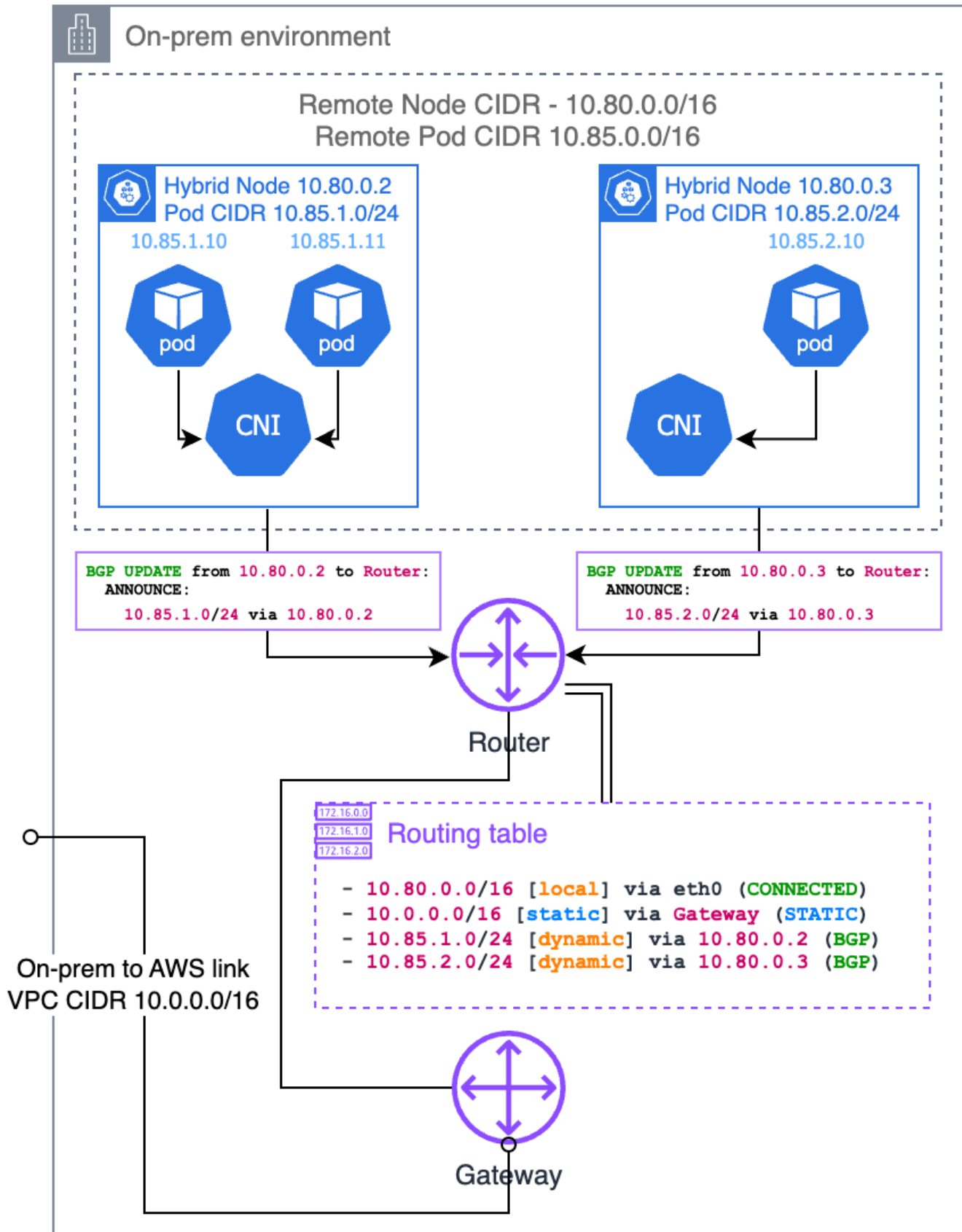
```
    namespace: default
    user: default
  name: default
  current-context: default
  users:
  - name: default
    user:
      tokenFile: /var/run/secrets/kubernetes.io/serviceaccount/token
kind: ConfigMap
metadata:
  name: kube-proxy
  namespace: kube-system
```

Pod jarak jauh yang dapat dirutekan CIDRs

[the section called “Konsep jaringan”](#) Halaman ini merinci persyaratan untuk menjalankan webhook pada node hybrid atau agar pod berjalan di cloud node berkomunikasi dengan pod yang berjalan pada node hybrid. Persyaratan utamanya adalah router lokal perlu mengetahui node mana yang bertanggung jawab atas IP pod tertentu. Ada beberapa cara untuk mencapai hal ini, termasuk Border Gateway Protocol (BGP), rute statis, dan proxy Address Resolution Protocol (ARP). Ini tercakup dalam bagian berikut.

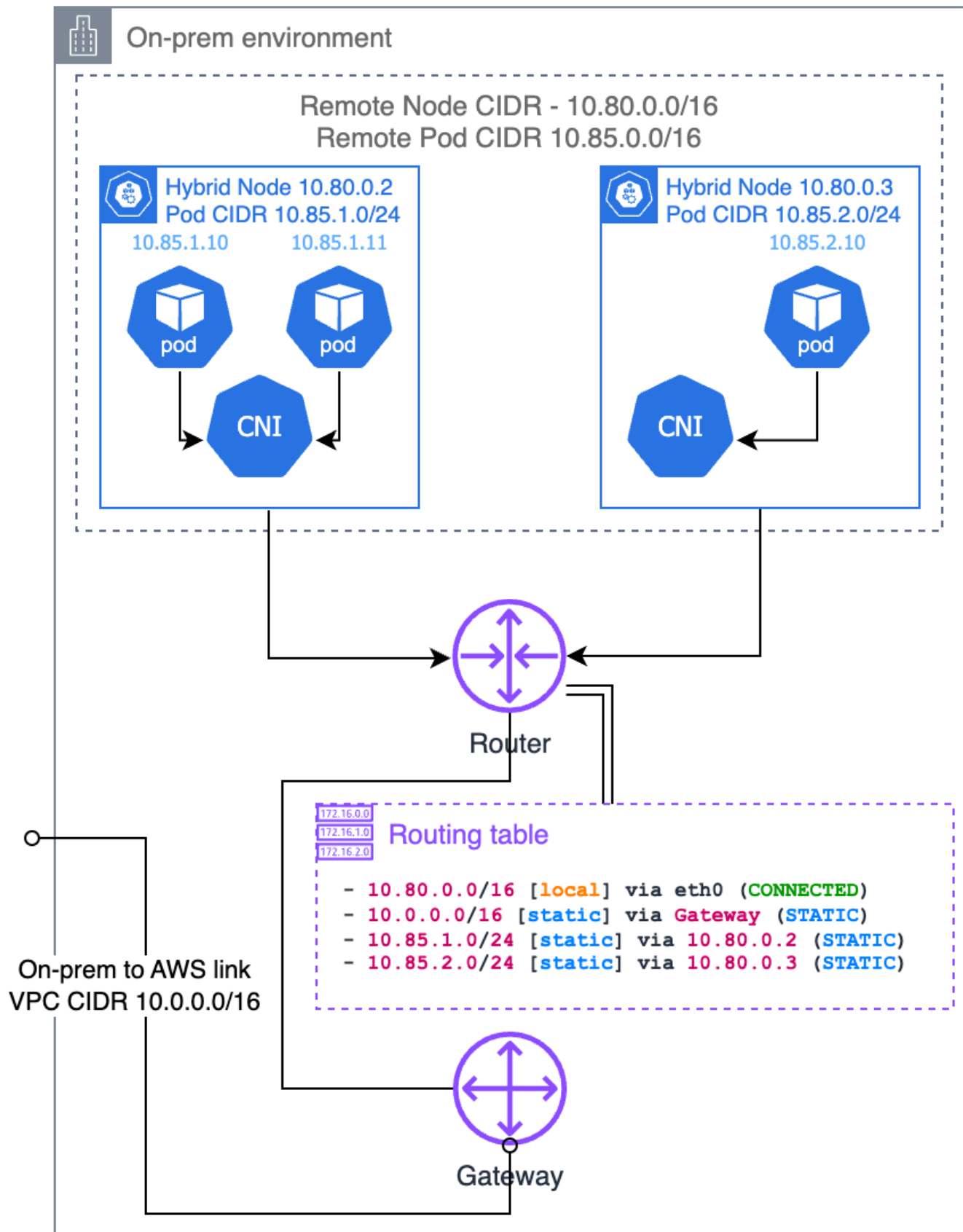
Protokol Gerbang Perbatasan (BGP)

Jika CNI Anda mendukungnya (seperti Cilium dan Calico), Anda dapat menggunakan mode BGP CNI Anda untuk menyebarkan rute ke pod CIDRs per node Anda dari node Anda ke router lokal Anda. Saat menggunakan mode BGP CNI, CNI Anda bertindak sebagai router virtual, sehingga router lokal Anda berpikir pod CIDR milik subnet yang berbeda dan node Anda adalah gateway ke subnet itu.



Rute statis

Atau, Anda dapat mengonfigurasi rute statis di router lokal Anda. Ini adalah cara paling sederhana untuk merutekan pod CIDR lokal ke VPC Anda, tetapi ini juga yang paling rawan kesalahan dan sulit dipertahankan. Anda perlu memastikan bahwa rute selalu up-to-date dengan node yang ada dan pod yang ditetapkan CIDRs. Jika jumlah node Anda kecil dan infrastruktur statis, ini adalah opsi yang layak dan menghilangkan kebutuhan akan dukungan BGP di router Anda. Jika Anda memilih ini, kami sarankan untuk mengonfigurasi CNI Anda dengan irisan pod CIDR yang ingin Anda tetapkan ke setiap node alih-alih membiarkan IPAM-nya memutuskan.



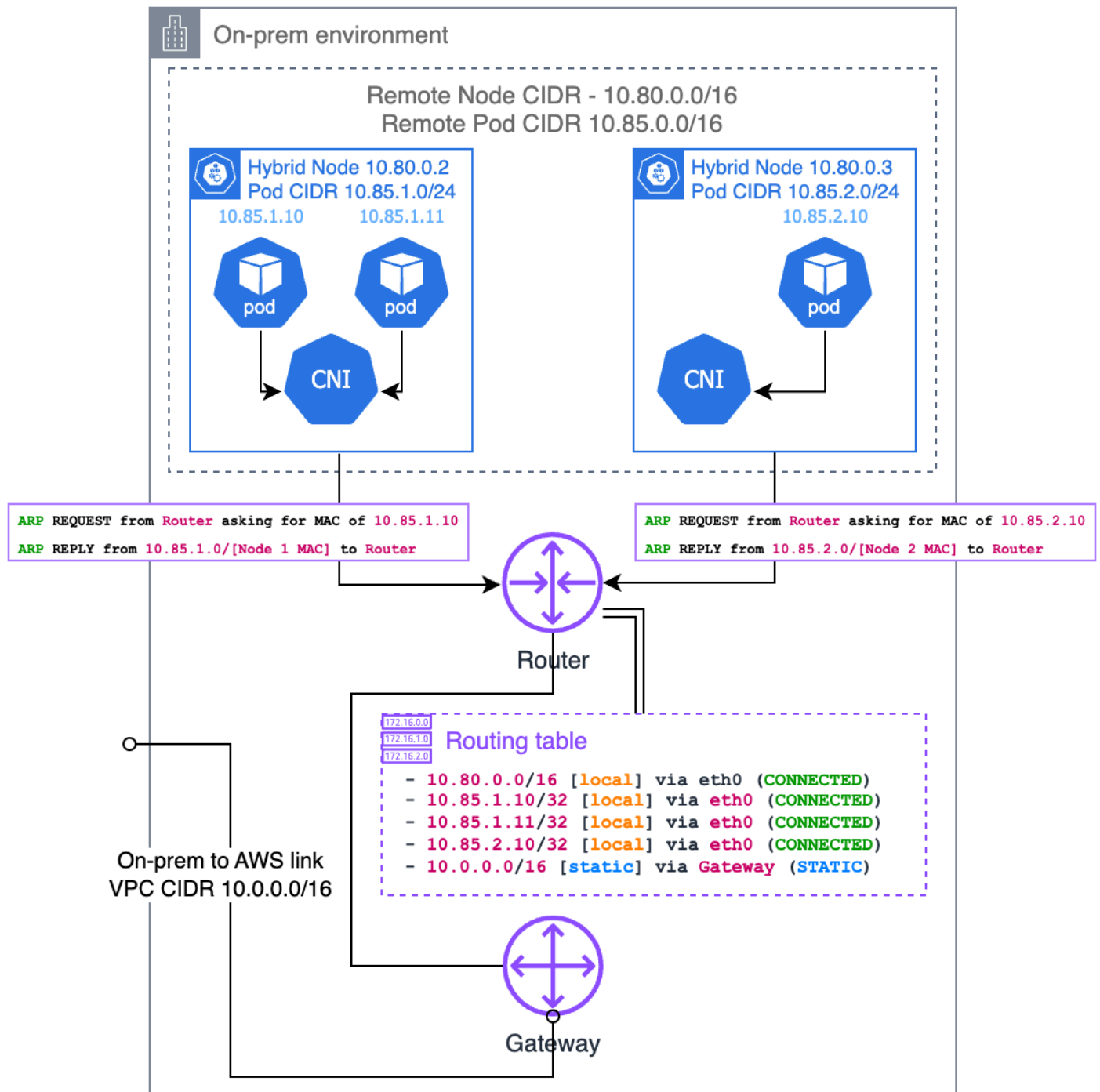
Proksi Protokol Resolusi Alamat (ARP)

Proxy ARP adalah pendekatan lain untuk membuat pod lokal IPs dapat dirutekan, sangat berguna ketika node hybrid Anda berada di jaringan Layer 2 yang sama dengan router lokal Anda. Dengan proksi ARP diaktifkan, sebuah node merespons permintaan ARP untuk pod yang dihost-nya, meskipun IPs itu IPs milik subnet yang berbeda.

Ketika perangkat di jaringan lokal Anda mencoba untuk mencapai IP pod, pertama-tama ia mengirimkan permintaan ARP yang menanyakan “Siapa yang memiliki IP ini?”. Node hybrid hosting pod itu akan merespons dengan alamat MAC-nya sendiri, mengatakan “Saya dapat menangani lalu lintas untuk IP itu.” Ini menciptakan jalur langsung antara perangkat di jaringan lokal Anda dan pod tanpa memerlukan konfigurasi router.

Agar ini berfungsi, CNI Anda harus mendukung fungsionalitas ARP proxy. Cilium memiliki dukungan bawaan untuk ARP proxy yang dapat Anda aktifkan melalui konfigurasi. Pertimbangan utamanya adalah bahwa pod CIDR tidak boleh tumpang tindih dengan jaringan lain di lingkungan Anda, karena ini dapat menyebabkan konflik perutean.

Pendekatan ini memiliki beberapa keunggulan: * Tidak perlu mengkonfigurasi router Anda dengan BGP atau mempertahankan rute statis * Bekerja dengan baik di lingkungan di mana Anda tidak memiliki kendali atas konfigurasi router Anda



Pod-to-Pod enkapsulasi

Di lingkungan lokal, CNIs biasanya menggunakan protokol enkapsulasi untuk membuat jaringan overlay yang dapat beroperasi di atas jaringan fisik tanpa perlu mengkonfigurasi ulang. Bagian ini menjelaskan cara kerja enkapsulasi ini. Perhatikan bahwa beberapa detail mungkin berbeda tergantung pada CNI yang Anda gunakan.

Enkapsulasi membungkus paket jaringan pod asli di dalam paket jaringan lain yang dapat dirutekan melalui jaringan fisik yang mendasarinya. Hal ini memungkinkan pod untuk berkomunikasi di seluruh node yang menjalankan CNI yang sama tanpa memerlukan jaringan fisik untuk mengetahui cara merutekan pod CIDRs tersebut.

Protokol enkapsulasi yang paling umum digunakan dengan Kubernetes adalah Virtual Extensible LAN (VXLAN), meskipun yang lain (seperti Geneve) juga tersedia tergantung pada CNI Anda.

Enkapsulasi VXLAN

VXLAN merangkum frame Ethernet Layer 2 dalam paket UDP. Ketika sebuah pod mengirimkan lalu lintas ke pod lain pada node yang berbeda, CNI melakukan hal berikut:

1. CNI mencegat paket dari Pod A
2. CNI membungkus paket asli dalam header VXLAN
3. Paket yang dibungkus ini kemudian dikirim melalui tumpukan jaringan reguler node ke node tujuan
4. CNI pada node tujuan membuka bungkus paket dan mengirimkannya ke Pod B

Inilah yang terjadi pada struktur paket selama enkapsulasi VXLAN:

Pod-to-Pod Paket Asli:

```
+-----+-----+-----+-----+
| Ethernet Header | IP Header   | TCP/UDP    | Payload     |
| Src: Pod A MAC  | Src: Pod A IP | Src Port   |             |
| Dst: Pod B MAC  | Dst: Pod B IP | Dst Port   |             |
+-----+-----+-----+-----+
```

Setelah Enkapsulasi VXLAN:

```
+-----+-----+-----+-----+
+-----+
| Outer Ethernet | Outer IP    | Outer UDP  | VXLAN       | Original Pod-to-Pod
|               |             |            |             |
| Src: Node A MAC | Src: Node A | Src: Random | VNI: xx    | Packet (unchanged
|               |             |            |             |
| Dst: Node B MAC | Dst: Node B | Dst: 4789   |             | from above)
|               |             |            |             |
+-----+-----+-----+-----+
+-----+
```

VXLAN Network Identifier (VNI) membedakan antara jaringan overlay yang berbeda.

Skenario komunikasi pod

Pod pada node hybrid yang sama

Ketika pod pada node hibrida yang sama berkomunikasi, biasanya tidak diperlukan enkapsulasi. CNI mengatur rute lokal yang mengarahkan lalu lintas antar pod melalui antarmuka virtual internal node:

```
Pod A -> veth0 -> node's bridge/routing table -> veth1 -> Pod B
```

Paket tidak pernah meninggalkan node dan tidak memerlukan enkapsulasi.

Pod pada node hibrida yang berbeda

Komunikasi antar pod pada node hybrid yang berbeda membutuhkan enkapsulasi:

```
Pod A -> CNI -> [VXLAN encapsulation] -> Node A network -> router or gateway -> Node B  
network -> [VXLAN decapsulation] -> CNI -> Pod B
```

Hal ini memungkinkan lalu lintas pod untuk melintasi infrastruktur jaringan fisik tanpa memerlukan jaringan fisik untuk memahami routing IP pod.

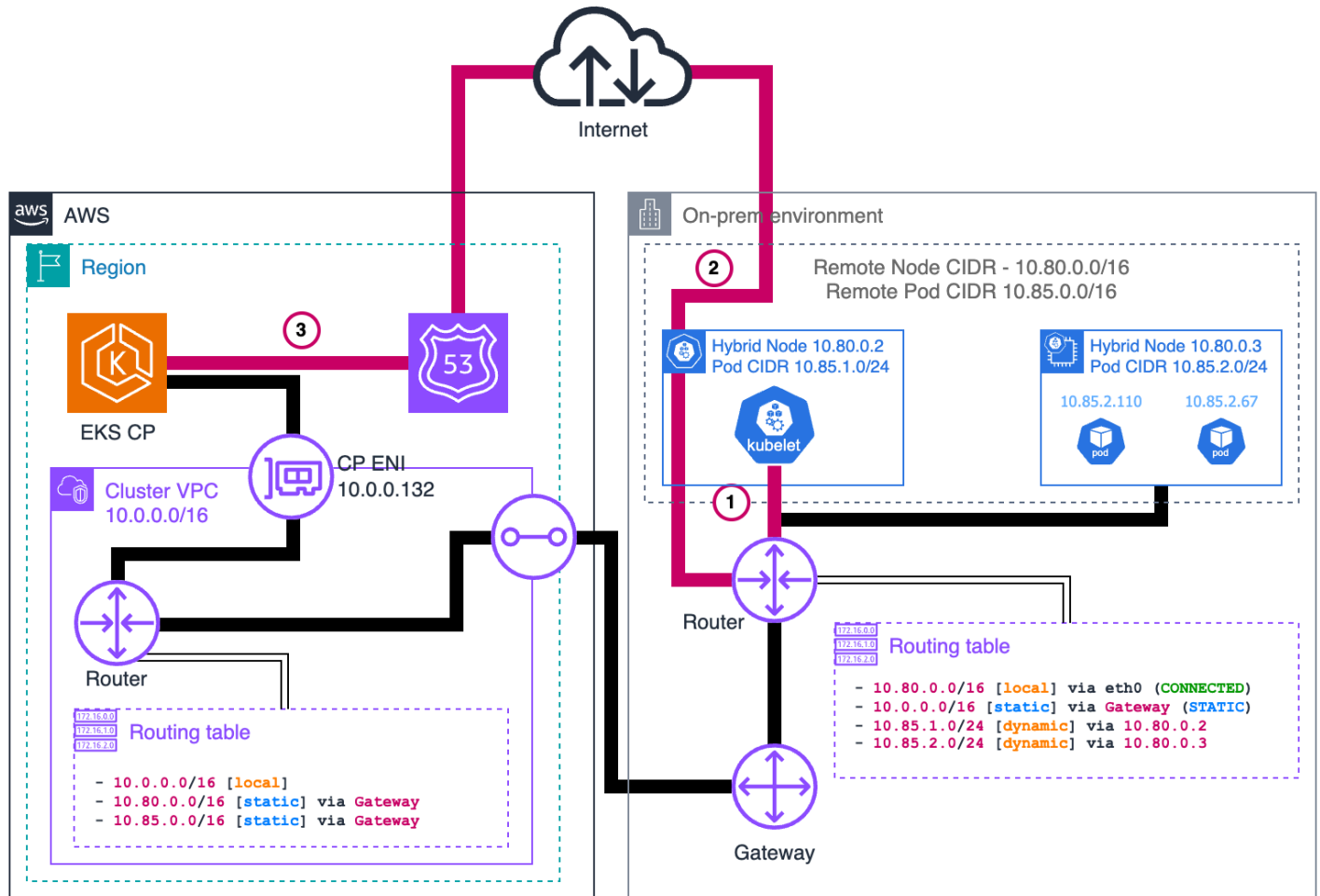
Arus lalu lintas jaringan untuk node hibrida

Halaman ini merinci arus lalu lintas jaringan untuk EKS Hybrid Nodes dengan diagram yang menunjukkan jalur end-to-end jaringan untuk jenis lalu lintas yang berbeda.

Arus lalu lintas berikut tercakup:

- [the section called “Node hibrida kubelet ke bidang kontrol EKS”](#)
- [the section called “Pesawat kontrol EKS ke node hybrid \(kubeletserver\)”](#)
- [the section called “Pod berjalan pada node hibrid ke bidang kontrol EKS”](#)
- [the section called “Pesawat kontrol EKS ke pod yang berjalan pada node hybrid \(webhooks\)”](#)
- [the section called “Pod-to-Pod berjalan pada node hybrid”](#)
- [the section called “Pod pada node cloud ke pod pada node hibrid \(lalu lintas timur-barat\)”](#)

Node hibrida **kubelet** ke bidang kontrol EKS



Permintaan

1. **kubelet** Memulai Permintaan

Ketika **kubelet** pada node hybrid perlu berkomunikasi dengan bidang kontrol EKS (misalnya, untuk melaporkan status node atau mendapatkan spesifikasi pod), ia menggunakan **kubeconfig** file yang disediakan selama pendaftaran node. Ini **kubeconfig** memiliki URL titik akhir server API (nama DNS Route53) daripada alamat IP langsung.

kubelet Melakukan pencarian DNS untuk titik akhir (misalnya,). <https://xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.gr7.us-west-2.eks.amazonaws.com> Dalam kluster akses publik, ini menyelesaikan ke alamat IP publik (katakanlah **54.239.118.52**) milik layanan EKS yang berjalan di. **AWSkubelet** Kemudian membuat permintaan HTTPS aman ke titik akhir ini. Paket awal terlihat seperti ini:


```

+-----+-----+-----+
| IP Header | TCP Header | Payload |
| Src: 10.80.0.2 | Src: 52390 (random) | |
| Dst: 54.239.118.52 | Dst: 443 | |
+-----+-----+-----+

```

2. Perutean Router Lokal

Karena IP tujuan adalah alamat IP publik dan bukan bagian dari jaringan lokal, paket ini akan kubelet dikirim ke gateway defaultnya (router lokal lokal). Router memeriksa IP tujuan dan menentukan itu adalah alamat IP publik.

Untuk lalu lintas publik, router biasanya meneruskan paket ke gateway internet atau router perbatasan yang menangani lalu lintas keluar ke internet. Ini dihilangkan dalam diagram dan akan tergantung pada bagaimana jaringan lokal Anda diatur. Paket ini melintasi infrastruktur jaringan lokal Anda dan akhirnya mencapai jaringan penyedia layanan internet Anda.

3. Pengiriman ke pesawat kontrol EKS

Paket ini berjalan melintasi internet publik dan jaringan transit sampai mencapai AWS jaringan. AWS jaringan merutekan paket ke titik akhir layanan EKS di wilayah yang sesuai. Ketika paket mencapai layanan EKS, itu diteruskan ke bidang kontrol EKS yang sebenarnya untuk cluster Anda.

Routing ini melalui internet publik berbeda dari jalur privat VPC yang akan kita lihat di arus lalu lintas lainnya. Perbedaan utamanya adalah ketika menggunakan mode akses publik, lalu lintas dari lokal kubelet (meskipun bukan dari pod) ke bidang kontrol EKS tidak melalui VPC Anda - ia menggunakan infrastruktur internet global sebagai gantinya.

Respons

Setelah bidang kontrol EKS memproses kubelet permintaan, ia mengirimkan respons kembali:

3. Pesawat kontrol EKS mengirimkan respons

Bidang kontrol EKS menciptakan paket respons. Paket ini memiliki IP publik sebagai sumber dan IP node hybrid sebagai tujuan:

```

+-----+-----+-----+
| IP Header | TCP Header | Payload |
| Src: 54.239.118.52 | Src: 443 | |
+-----+-----+-----+

```

Dst: 10.80.0.2	Dst: 52390		
+-----+	+-----+	+-----+	+-----+

2. Perutean Internet

Paket respons berjalan kembali melalui internet, mengikuti jalur perutean yang ditentukan oleh penyedia layanan internet, hingga mencapai router tepi jaringan lokal Anda.

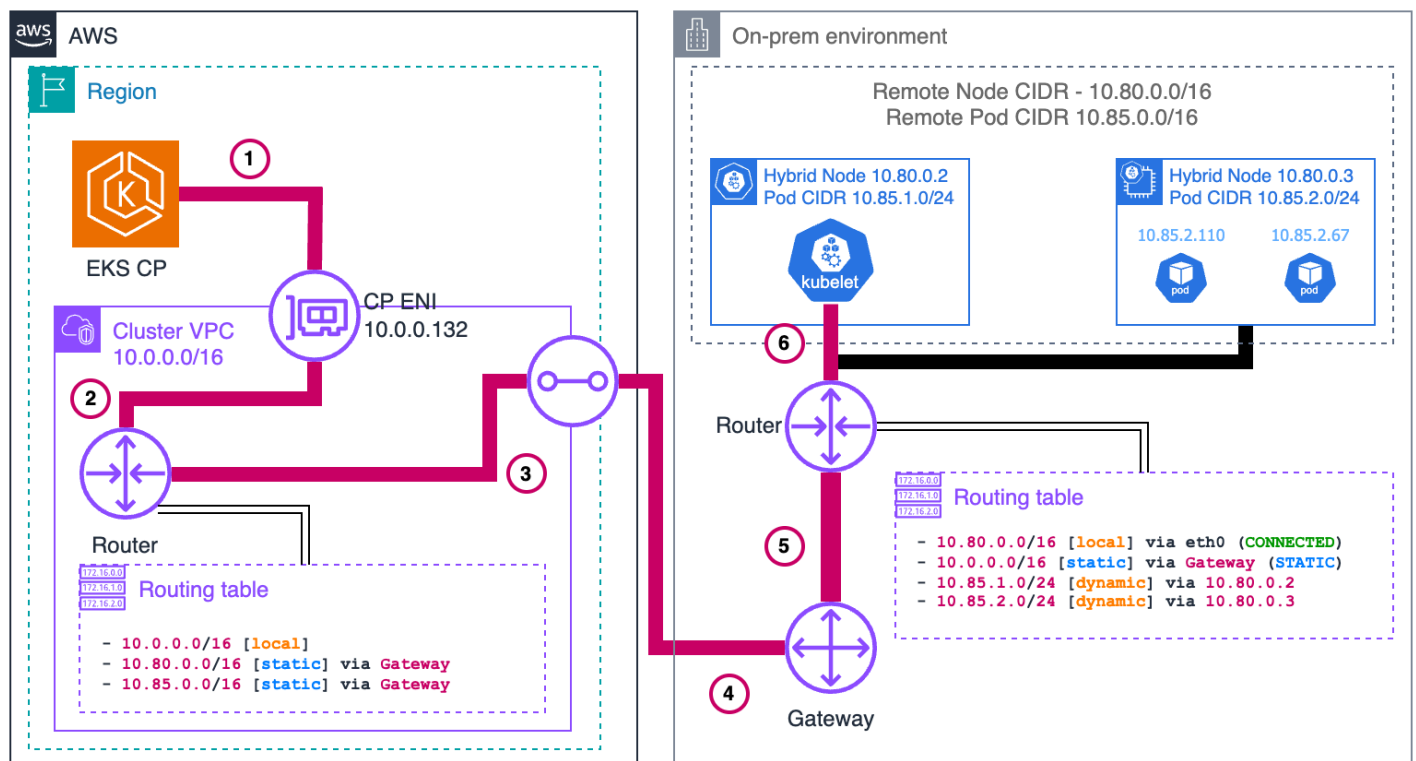
1. Pengiriman Lokal

Router lokal Anda menerima paket dan mengenali IP tujuan (10.80.0.2) sebagai milik jaringan lokal Anda. Ini meneruskan paket melalui infrastruktur jaringan lokal Anda sampai mencapai target node hybrid, di mana kubelet menerima dan memproses respons.

Node hibrida **kube-proxy** ke bidang kontrol EKS

Jika Anda mengaktifkan akses endpoint publik untuk cluster, lalu lintas kembali menggunakan internet publik. Lalu lintas ini berasal dari node hibrida ke bidang kontrol EKS dan mengikuti jalur yang sama dengan lalu lintas dari kubelet ke bidang kontrol EKS. kube-proxy

Pesawat kontrol EKS ke node hybrid (**kubeletserver**)



Permintaan

1. Server API EKS Kubernetes memulai permintaan

Server API EKS Kubernetes mengambil alamat IP node (10.80.0.2) dari status objek node. Kemudian merutekan permintaan ini melalui ENI-nya di VPC, karena IP tujuan milik node jarak jauh yang dikonfigurasi CIDR (). 10.80.0.0/16 Paket awal terlihat seperti ini:

```
+-----+-----+-----+
| IP Header | TCP Header | Payload |
| Src: 10.0.0.132 | Src: 67493 (random) | |
| Dst: 10.80.0.2 | Dst: 10250 | |
+-----+-----+-----+
```

2. Pemrosesan jaringan VPC

Paket meninggalkan ENI dan memasuki lapisan jaringan VPC, di mana ia diarahkan ke gateway subnet untuk perutean lebih lanjut.

3. Pencarian tabel rute VPC

Tabel rute VPC untuk subnet yang berisi bidang kontrol EKS ENI memiliki rute tertentu (yang kedua dalam diagram) untuk simpul jarak jauh CIDR. Berdasarkan aturan routing ini, paket diarahkan ke gateway. VPC-to-onprem

4. Transit lintas batas

Gateway mentransfer paket melintasi batas cloud melalui koneksi yang Anda buat (seperti Direct Connect atau VPN) ke jaringan lokal Anda.

5. Penerimaan jaringan lokal

Paket tiba di router lokal lokal Anda yang menangani lalu lintas untuk subnet tempat node hybrid Anda berada.

6. Pengiriman akhir

Router lokal mengidentifikasi bahwa alamat IP (10.80.0.2) tujuan milik jaringan yang terhubung langsung dan meneruskan paket langsung ke node hibrida target, di mana kubelet menerima dan memproses permintaan.

Respons

Setelah node hybrid kubelet memproses permintaan, ia mengirimkan kembali respons mengikuti jalur yang sama secara terbalik:

```
+-----+-----+-----+
| IP Header | TCP Header | Payload |
| Src: 10.80.0.2 | Src: 10250 |         |
| Dst: 10.0.0.132 | Dst: 67493 |         |
+-----+-----+-----+
```

6. kubelet Mengirim Respon

Pada node hybrid (10.80.0.2) membuat paket respons dengan IP sumber asli sebagai tujuan. Tujuan bukan milik jaringan lokal sehingga dikirim ke gateway default host, yang merupakan router lokal.

5. Perutean Router Lokal

Router menentukan bahwa tujuan IP (10.0.0.132) milik 10.0.0.0/16, yang memiliki rute yang menunjuk ke gateway yang terhubung ke AWS.

4. Pengembalian Lintas Batas

Paket berjalan kembali melalui koneksi lokal ke VPC yang sama (seperti Direct Connect atau VPN), melintasi batas cloud dalam arah sebaliknya.

3. Perutean VPC

Ketika paket tiba di VPC, tabel rute mengidentifikasi bahwa IP tujuan milik CIDR VPC. Rute paket dalam VPC.

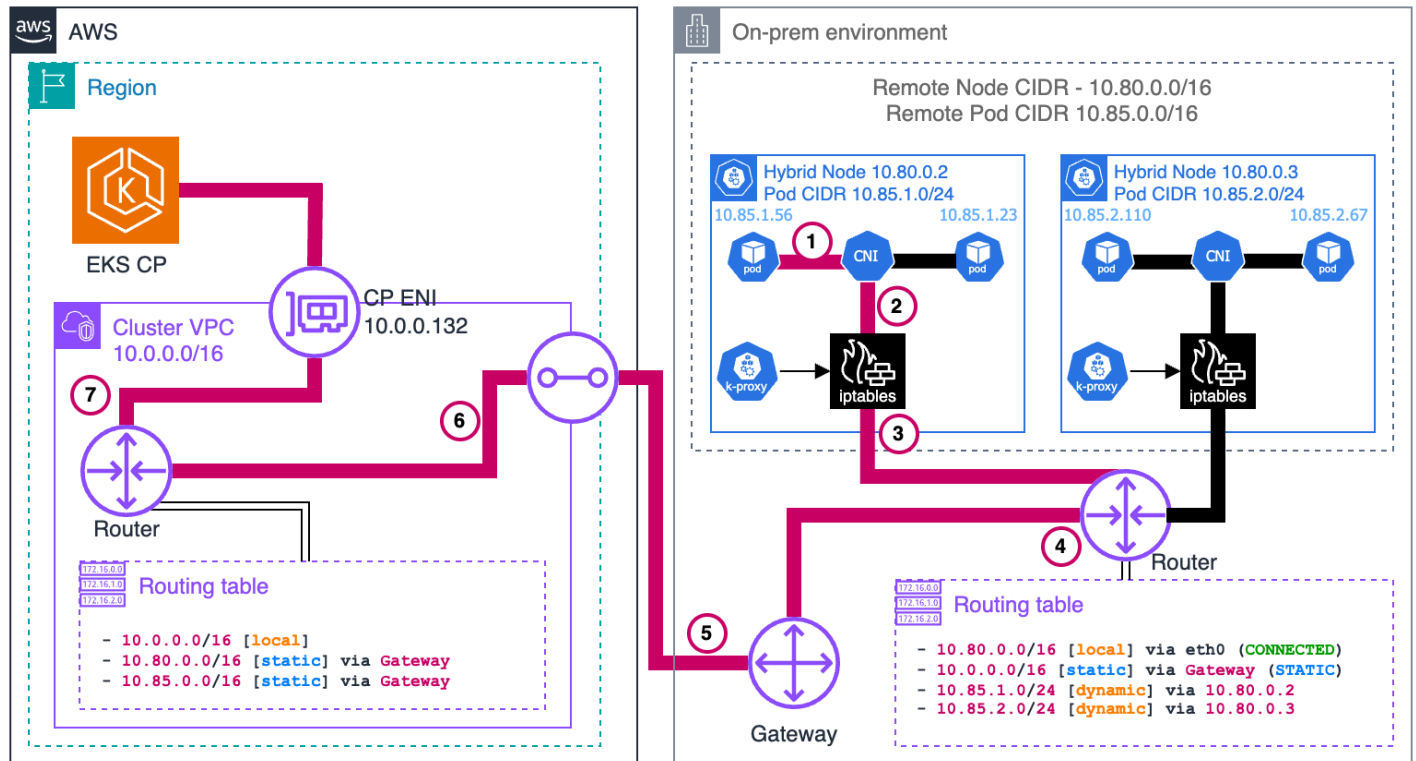
2. Pengiriman Jaringan VPC

Lapisan jaringan VPC meneruskan paket ke subnet dengan bidang kontrol EKS ENI (). 10.0.0.132

1. Penerimaan ENI

Paket mencapai pesawat kontrol EKS ENI yang terpasang ke server API Kubernetes, menyelesaikan perjalanan pulang pergi.

Pod berjalan pada node hibrid ke bidang kontrol EKS



Tanpa CNI NAT

Permintaan

Pod umumnya berbicara dengan server API Kubernetes melalui layanan. kubernetes IP layanan adalah IP pertama dari CIDR layanan cluster. Konvensi ini memungkinkan pod yang perlu dijalankan sebelum CoreDNS tersedia untuk mencapai server API, misalnya, CNI. Permintaan meninggalkan pod dengan IP layanan sebagai tujuan. Misalnya, jika layanan CIDR adalah 172.16.0.0/16, IP layanan akan 172.16.0.1.

1. Pod Memulai Permintaan

Pod mengirimkan permintaan ke kubernetes layanan IP (172.16.0.1) pada port server API (443) dari port sumber acak. Paketnya terlihat seperti ini:

```
+-----+-----+
| IP Header | TCP Header | Payload |
| Src: 10.85.1.56 | Src: 67493 (random) |
| Dst: 172.16.0.1 | Dst: 443 |
+-----+-----+
```

2. Pengolahan CNI

CNI mendeteksi bahwa IP tujuan bukan milik pod CIDR mana pun yang dikelolanya. Karena NAT keluar dinonaktifkan, CNI meneruskan paket ke tumpukan jaringan host tanpa memodifikasinya.

3. Pemrosesan Jaringan Node

Paket memasuki tumpukan jaringan node di mana `netfilter` hook memicu `iptables` aturan yang ditetapkan oleh `kube-proxy`. Beberapa aturan berlaku dalam urutan sebagai berikut:

1. Paket pertama mengenai KUBE-SERVICES rantai, yang berisi aturan yang cocok dengan ClusterIP dan port setiap layanan.
2. Aturan pencocokan melompat ke KUBE-SVC-XXX rantai untuk kubernetes layanan (paket yang ditujukan untuk `172.16.0.1:443`), yang berisi aturan penyeimbangan beban.
3. Aturan penyeimbangan beban secara acak memilih salah satu KUBE-SEP-XXX rantai untuk bidang kontrol ENI IPs (`10.0.0.132` atau `10.0.1.23`).
4. KUBE-SEP-XXX Rantai yang dipilih memiliki aturan aktual yang mengubah IP tujuan dari IP layanan ke IP yang dipilih. Ini disebut Destination Network Address Translation (DNAT).

Setelah aturan ini diterapkan, dengan asumsi bahwa IP ENI bidang kontrol EKS yang dipilih adalah `10.0.0.132`, paketnya terlihat seperti ini:

```
+-----+-----+-----+
| IP Header | TCP Header | Payload |
| Src: 10.85.1.56 | Src: 67493 (random) | |
| Dst: 10.0.0.132 | Dst: 443 | |
+-----+-----+-----+
```

Node meneruskan paket ke gateway defaultnya karena IP tujuan tidak ada di jaringan lokal.

4. Perutean Router Lokal

Router lokal menentukan bahwa IP tujuan (`10.0.0.132`) milik VPC CIDR (`10.0.0.0/16`) dan meneruskannya ke gateway yang terhubung ke AWS.

5. Transit Lintas Batas

Paket berjalan melalui koneksi yang Anda buat (seperti Direct Connect atau VPN) melintasi batas cloud ke VPC.

6. Pengiriman Jaringan VPC

Lapisan jaringan VPC merutekan paket ke subnet yang benar di mana bidang kontrol EKS ENI () 10.0.0.132 berada.

7. Penerimaan ENI

Paket mencapai bidang kontrol EKS ENI yang terpasang ke server API Kubernetes.

Respons

Setelah bidang kontrol EKS memproses permintaan, ia mengirimkan respons kembali ke pod:

7. Server API Mengirim Respons

Server API EKS Kubernetes membuat paket respons dengan IP sumber asli sebagai tujuan. Paketnya terlihat seperti ini:

```
+-----+-----+-----+
| IP Header | TCP Header | Payload |
| Src: 10.0.0.132 | Src: 443 | |
| Dst: 10.85.1.56 | Dst: 67493 | |
+-----+-----+-----+
```

Karena IP tujuan milik pod jarak jauh yang dikonfigurasi CIDR (10.85.0.0/16), ia mengirimkannya melalui ENI di VPC dengan router subnet sebagai hop berikutnya.

6. Perutean VPC

Tabel rute VPC berisi entri untuk pod jarak jauh CIDR (10.85.0.0/16), mengarahkan lalu lintas ini ke gateway. VPC-to-onprem

5. Transit Lintas Batas

Gateway mentransfer paket melintasi batas cloud melalui koneksi yang Anda buat (seperti Direct Connect atau VPN) ke jaringan lokal Anda.

4. Penerimaan Jaringan Lokal

Paket tiba di router lokal lokal Anda.

3. Pengiriman ke node

Tabel router memiliki entri untuk `10.85.1.0/24` dengan `10.80.0.2` sebagai hop berikutnya, mengirimkan paket ke node kami.

2. Pemrosesan Jaringan Node

Karena paket diproses oleh tumpukan jaringan node, `conntrack` (bagian dari `netfilter`) mencocokkan paket dengan koneksi yang awalnya dibuat oleh pod. Karena DNAT awalnya diterapkan, `conntrack` membalikkan DNAT dengan menulis ulang IP sumber dari IP ENI bidang kontrol EKS ke IP layanan: `kubernetes`

```
+-----+-----+-----+
| IP Header | TCP Header | Payload |
| Src: 172.16.0.1 | Src: 443 | |
| Dst: 10.85.1.56 | Dst: 67493 | |
+-----+-----+-----+
```

1. Pengolahan CNI

CNI mengidentifikasi bahwa IP tujuan milik sebuah pod dalam jaringannya dan mengirimkan paket ke namespace jaringan pod yang benar.

Alur ini menunjukkan mengapa Remote Pod CIDRs harus dapat dirutekan dengan benar dari VPC sampai ke node tertentu yang menghosting setiap pod - seluruh jalur pengembalian bergantung pada perutean pod yang tepat di jaringan cloud dan IPs lokal.

Dengan CNI NAT

Aliran ini sangat mirip dengan yang tanpa CNI NAT, tetapi dengan satu perbedaan utama: CNI menerapkan sumber NAT (SNAT) ke paket sebelum mengirimnya ke tumpukan jaringan node. Ini mengubah IP sumber paket ke IP node, memungkinkan paket untuk dirutekan kembali ke node tanpa memerlukan konfigurasi routing tambahan.

Permintaan

1. Pod Memulai Permintaan

Pod mengirimkan permintaan ke IP `kubernetes` layanan (`172.16.0.1`) pada port server API EKS `Kubernetes` (443) dari port sumber acak. Paketnya terlihat seperti ini:

```
+-----+-----+-----+
| IP Header | TCP Header | Payload |
| Src: 10.85.1.56 | Src: 67493 (random) | |
+-----+-----+-----+
```



```
| Dst: 172.16.0.1 | Dst: 443 | |
+-----+-----+-----+
```

2. Pengolahan CNI

CNI mendeteksi bahwa IP tujuan bukan milik pod CIDR mana pun yang dikelolanya. Karena NAT keluar diaktifkan, CNI menerapkan SNAT ke paket, mengubah IP sumber ke IP node sebelum meneruskannya ke tumpukan jaringan node:

```
+-----+-----+-----+
| IP Header | TCP Header | Payload |
| Src: 10.80.0.2 | Src: 67493 (random) | |
| Dst: 172.16.0.1 | Dst: 443 | |
+-----+-----+-----+
```

Catatan: CNI dan iptables ditampilkan dalam contoh sebagai blok terpisah untuk kejelasan, tetapi dalam praktiknya, ada kemungkinan beberapa CNIs penggunaan iptables untuk menerapkan NAT.

3. Pemrosesan Jaringan Node

Di sini iptables aturan yang ditetapkan oleh kube-proxy berperilaku sama seperti pada contoh sebelumnya, load balancing paket ke salah satu bidang kontrol EKS. ENIs Paket sekarang terlihat seperti ini:

```
+-----+-----+-----+
| IP Header | TCP Header | Payload |
| Src: 10.80.0.2 | Src: 67493 (random) | |
| Dst: 10.0.0.132 | Dst: 443 | |
+-----+-----+-----+
```

Node meneruskan paket ke gateway defaultnya karena IP tujuan tidak ada di jaringan lokal.

4. Perutean Router Lokal

Router lokal menentukan bahwa IP tujuan (10.0.0.132) milik VPC CIDR (10.0.0.0/16) dan meneruskannya ke gateway yang terhubung ke AWS

5. Transit Lintas Batas

Paket berjalan melalui koneksi yang Anda buat (seperti Direct Connect atau VPN) melintasi batas cloud ke VPC.

6. Pengiriman Jaringan VPC

Lapisan jaringan VPC merutekan paket ke subnet yang benar di mana bidang kontrol EKS ENI () `10.0.0.132` berada.

7. Penerimaan ENI

Paket mencapai bidang kontrol EKS ENI yang terpasang ke server API Kubernetes.

Respons

Setelah bidang kontrol EKS memproses permintaan, ia mengirimkan respons kembali ke pod:

7. Server API Mengirim Respons

Server API EKS Kubernetes membuat paket respons dengan IP sumber asli sebagai tujuan. Paketnya terlihat seperti ini:

```
+-----+-----+-----+
| IP Header | TCP Header | Payload |
| Src: 10.0.0.132 | Src: 443 | |
| Dst: 10.80.0.2 | Dst: 67493 | |
+-----+-----+-----+
```

Karena IP tujuan milik node jarak jauh yang dikonfigurasi CIDR (`10.80.0.0/16`), ia mengirimkannya melalui ENI-nya di VPC dengan router subnet sebagai hop berikutnya.

6. Perutean VPC

Tabel rute VPC berisi entri untuk node jarak jauh CIDR (`10.80.0.0/16`), mengarahkan lalu lintas ini ke gateway. VPC-to-onprem

5. Transit Lintas Batas

Gateway mentransfer paket melintasi batas cloud melalui koneksi yang Anda buat (seperti Direct Connect atau VPN) ke jaringan lokal Anda.

4. Penerimaan Jaringan Lokal

Paket tiba di router lokal lokal Anda.

3. Pengiriman ke node

Router lokal mengidentifikasi bahwa alamat IP (10.80.0.2) tujuan milik jaringan yang terhubung langsung dan meneruskan paket langsung ke node hibrida target.

2. Pemrosesan Jaringan Node

Karena paket diproses oleh tumpukan jaringan node, `conntrack` (bagian dari `netfilter`) mencocokkan paket dengan koneksi yang awalnya dibuat oleh pod dan karena DNAT awalnya diterapkan, itu membalikkan ini dengan menulis ulang IP sumber dari IP ENI bidang kontrol EKS ke IP layanan: `kubernetes`

```
+-----+-----+-----+
| IP Header | TCP Header | Payload |
| Src: 172.16.0.1 | Src: 443 | |
| Dst: 10.80.0.2 | Dst: 67493 | |
+-----+-----+-----+
```

1. Pengolahan CNF

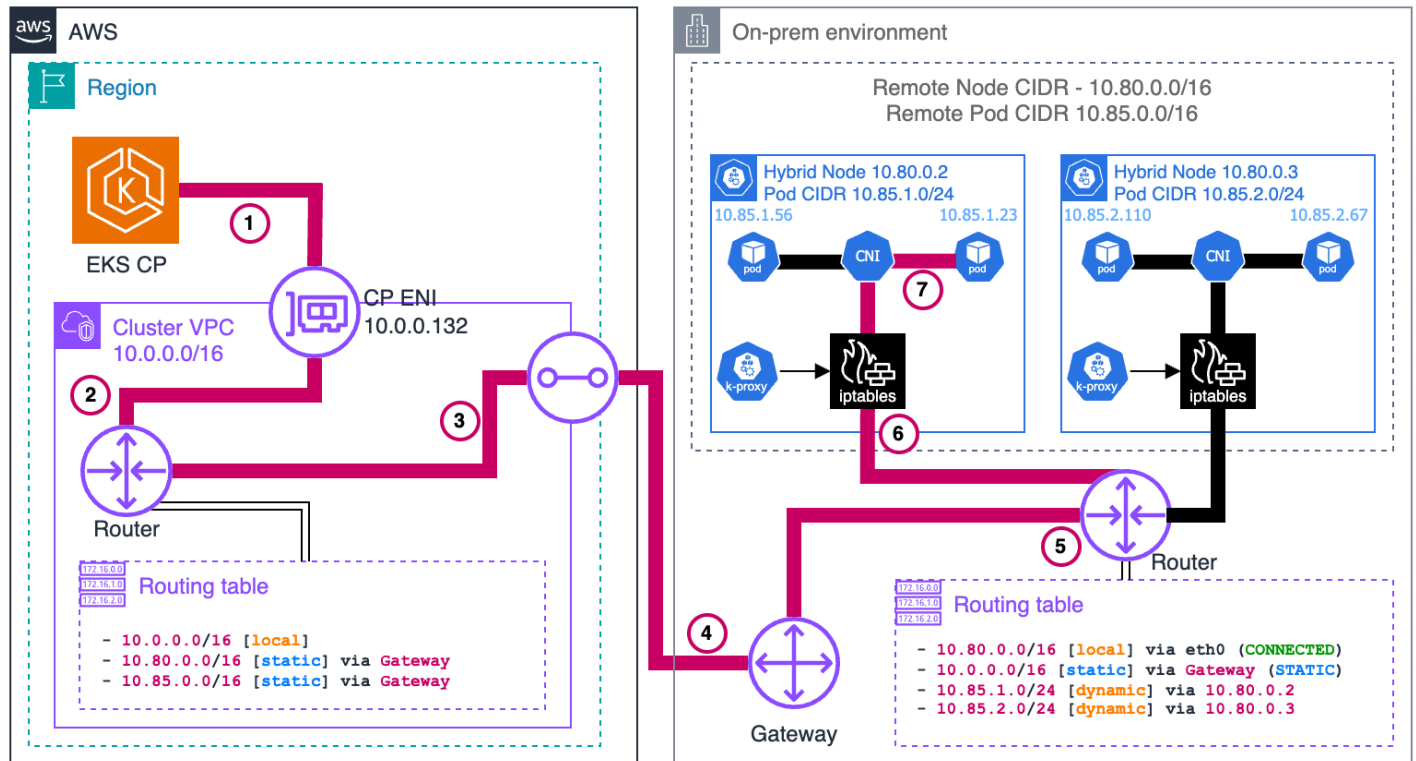
CNF mengidentifikasi paket ini milik koneksi di mana sebelumnya telah menerapkan SNAT. Ini membalikkan SNAT, mengubah IP tujuan kembali ke IP pod:

```
+-----+-----+-----+
| IP Header | TCP Header | Payload |
| Src: 172.16.0.1 | Src: 443 | |
| Dst: 10.85.1.56 | Dst: 67493 | |
+-----+-----+-----+
```

CNF mendeteksi IP tujuan milik sebuah pod dalam jaringannya dan mengirimkan paket ke namespace jaringan pod yang benar.

Alur ini menampilkan bagaimana CNF NAT-ing dapat menyederhanakan konfigurasi dengan memungkinkan paket dirutekan kembali ke node tanpa memerlukan perutean tambahan untuk pod. CIDRs

Pesawat kontrol EKS ke pod yang berjalan pada node hybrid (webhooks)



Pola lalu lintas ini paling sering terlihat dengan webhooks, di mana bidang kontrol EKS perlu langsung memulai koneksi ke server webhook yang berjalan di pod pada node hybrid. Contohnya termasuk memvalidasi dan memutasi webhook masuk, yang dipanggil oleh server API selama validasi sumber daya atau proses mutasi.

Permintaan

1. Server API EKS Kubernetes memulai permintaan

Ketika webhook dikonfigurasi di cluster dan operasi API yang relevan memicunya, server API EKS Kubernetes perlu membuat koneksi langsung ke pod server webhook. Server API pertama-tama mencari alamat IP pod dari sumber daya Service atau Endpoint yang terkait dengan webhook.

Dengan asumsi pod webhook berjalan pada node hybrid dengan IP 10.85.1.23, server API EKS Kubernetes membuat permintaan HTTPS ke endpoint webhook. Paket awal dikirim melalui bidang kontrol EKS ENI di VPC Anda karena 10.85.1.23 IP tujuan milik pod jarak jauh yang dikonfigurasi 10.85.0.0/16 CIDR (). Paketnya terlihat seperti ini:

```
+-----+-----+-----+
```

IP Header	TCP Header	Payload	
Src: 10.0.0.132	Src: 41892 (random)		
Dst: 10.85.1.23	Dst: 8443		
+-----+-----+-----+			

2. Pemrosesan Jaringan VPC

Paket meninggalkan bidang kontrol EKS ENI dan memasuki lapisan jaringan VPC dengan router subnet sebagai lompatan berikutnya.

3. Pencarian Tabel Rute VPC

Tabel rute VPC untuk subnet yang berisi bidang kontrol EKS ENI berisi rute khusus untuk pod jarak jauh CIDR (). 10.85.0.0/16 Aturan routing ini mengarahkan paket ke VPC-to-onprem gateway (misalnya, Virtual Private Gateway untuk Direct Connect atau koneksi VPN):

Destination	Target
10.0.0.0/16	local
10.85.0.0/16	vgw-id (VPC-to-onprem gateway)

4. Transit Lintas Batas

Gateway mentransfer paket melintasi batas cloud melalui koneksi yang Anda buat (seperti Direct Connect atau VPN) ke jaringan lokal Anda. Paket ini mempertahankan sumber asli dan alamat IP tujuan saat melintasi koneksi ini.

5. Penerimaan Jaringan Lokal

Paket tiba di router lokal lokal Anda. Router berkonsultasi dengan tabel peruteannya untuk menentukan cara mencapai alamat 10.85.1.23. Agar ini berfungsi, jaringan lokal Anda harus memiliki rute untuk pod CIDRs yang mengarahkan lalu lintas ke node hibrid yang sesuai.

Dalam hal ini, tabel rute router berisi entri yang menunjukkan bahwa 10.85.1.0/24 subnet dapat dijangkau melalui node hybrid dengan IP: 10.80.0.2

Destination	Next Hop
10.85.1.0/24	10.80.0.2

6. Pengiriman ke node

Berdasarkan entri tabel routing, router meneruskan paket ke node hybrid (). `10.80.0.2` Ketika paket tiba di node, itu terlihat sama seperti ketika server EKS Kubernetes API mengirimnya, dengan IP tujuan masih menjadi IP pod.

7. Pengolahan CNI

Tumpukan jaringan node menerima paket dan, melihat bahwa IP tujuan bukan IP node itu sendiri, meneruskannya ke CNI untuk diproses. CNI mengidentifikasi bahwa IP tujuan milik pod yang berjalan secara lokal pada node ini dan meneruskan paket ke pod yang benar melalui antarmuka virtual yang sesuai:

```
Original packet -> node routing -> CNI -> Pod's network namespace
```

Server webhook di pod menerima permintaan dan memprosesnya.

Respons

Setelah pod webhook memproses permintaan, ia mengirimkan kembali respons mengikuti jalur yang sama secara terbalik:

7. Pod Mengirim Respons

Pod webhook membuat paket respons dengan IP sendiri sebagai sumber dan pemohon asli (ENI bidang kontrol EKS) sebagai tujuan:

```
+-----+-----+-----+
| IP Header | TCP Header | Payload |
| Src: 10.85.1.23 | Src: 8443 | |
| Dst: 10.0.0.132 | Dst: 41892 | |
+-----+-----+-----+
```

CNI mengidentifikasi bahwa paket ini masuk ke jaringan eksternal (bukan pod lokal) dan meneruskan paket ke tumpukan jaringan node dengan IP sumber asli yang dipertahankan.

6. Pemrosesan Jaringan Node

Node menentukan bahwa IP tujuan (`10.0.0.132`) tidak berada di jaringan lokal dan meneruskan paket ke gateway default (router lokal).

5. Perutean Router Lokal

Router lokal berkonsultasi dengan tabel routing dan menentukan bahwa IP tujuan (10.0.0.132) milik VPC CIDR (). 10.0.0.0/16 Ini meneruskan paket ke gateway yang terhubung ke AWS

4. Transit Lintas Batas

Paket berjalan kembali melalui koneksi lokal ke VPC yang sama, melintasi batas cloud dalam arah sebaliknya.

3. Perutean VPC

Ketika paket tiba di VPC, tabel rute mengidentifikasi bahwa IP tujuan milik subnet dalam VPC. Paket dirutekan sesuai di dalam VPC.

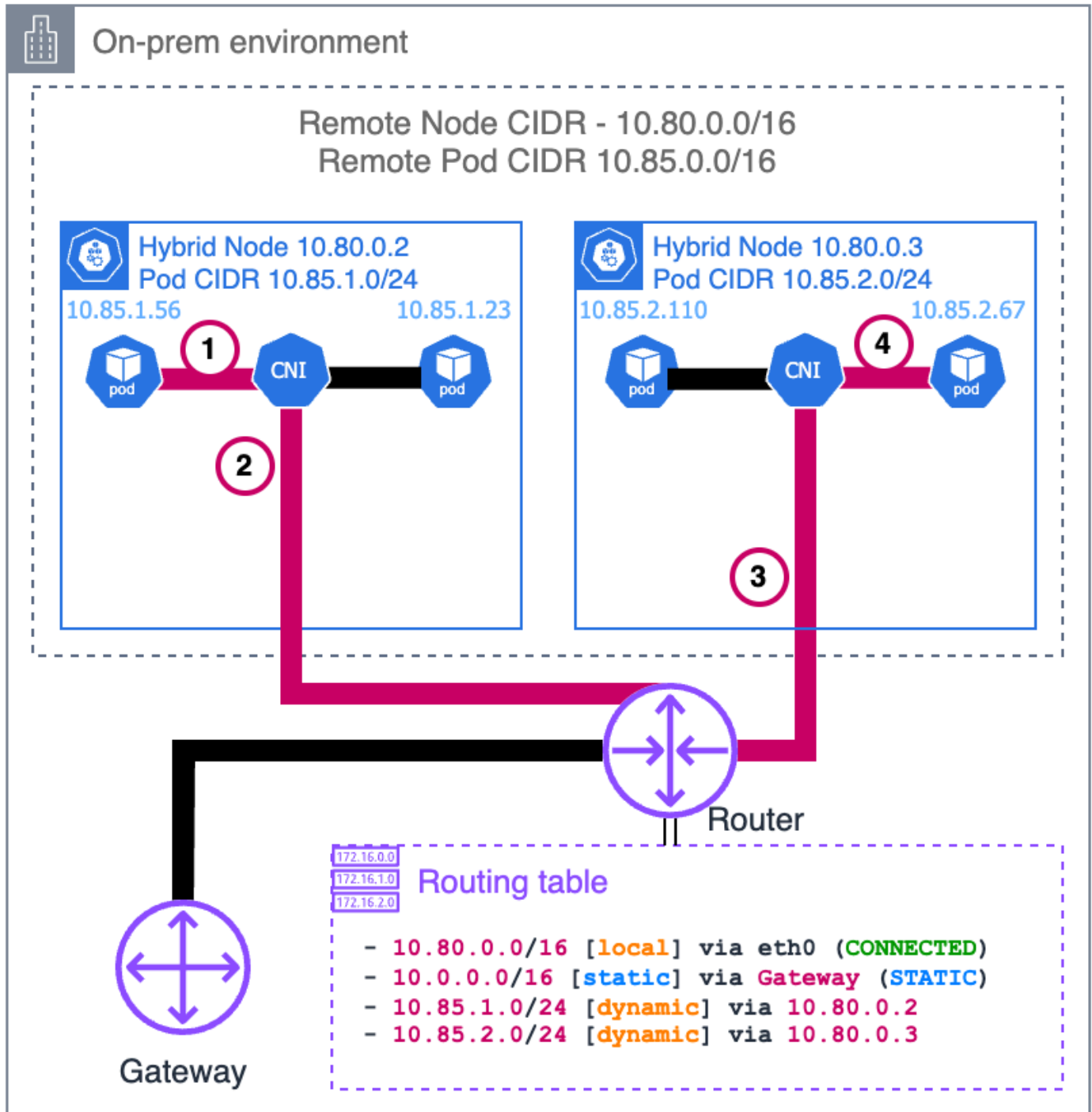
2. dan 1. EKS pesawat kontrol ENI Penerimaan

Paket mencapai ENI yang terpasang ke server API EKS Kubernetes, menyelesaikan perjalanan pulang pergi. Server API menerima respons webhook dan terus memproses permintaan API asli berdasarkan respons ini.

Arus lalu lintas ini menunjukkan mengapa pod jarak jauh CIDRs harus dikonfigurasi dan dirutekan dengan benar:

- VPC harus memiliki rute untuk pod jarak jauh yang CIDRs mengarah ke gateway lokal
- Jaringan lokal Anda harus memiliki rute untuk pod CIDRs yang mengarahkan lalu lintas ke node tertentu yang menghosting pod tersebut
- Tanpa konfigurasi routing ini, webhook dan layanan serupa lainnya yang berjalan di pod pada node hybrid tidak akan dapat dijangkau dari bidang kontrol EKS.

Pod-to-Pod berjalan pada node hybrid



Bagian ini menjelaskan bagaimana pod yang berjalan pada node hybrid yang berbeda berkomunikasi satu sama lain. Contoh ini mengasumsikan CNI Anda menggunakan VXLAN untuk enkapsulasi, yang

umum untuk seperti Cilium atau Calico. CNIs Proses keseluruhan serupa untuk protokol enkapsulasi lainnya seperti Geneve atau. IP-in-IP

Permintaan

1. Pod A Memulai Komunikasi

Pod A (10.85.1.56) pada Node 1 ingin mengirim lalu lintas ke Pod B (10.85.2.67) di Node 2. Paket awal terlihat seperti ini:

```
+-----+-----+-----+-----+
| Ethernet Header | IP Header      | TCP/UDP      | Payload      |
| Src: Pod A MAC  | Src: 10.85.1.56 | Src: 43721   |              |
| Dst: Gateway MAC | Dst: 10.85.2.67 | Dst: 8080    |              |
+-----+-----+-----+-----+
```

2. CNI Mencegat dan Memproses Paket

Ketika paket Pod A meninggalkan namespace jaringannya, CNI mencegatnya. CNI berkonsultasi dengan tabel routing dan menentukan: - Destination IP (10.85.2.67) milik pod CIDR - IP ini bukan pada node lokal tetapi milik Node 2 (10.80.0.3) - Paket perlu dienkapsulasi dengan VXLAN.

Keputusan untuk merangkum sangat penting karena jaringan fisik yang mendasarinya tidak tahu cara merutekan pod CIDRs secara langsung - ia hanya tahu cara merutekan lalu lintas antar node. IPs

CNI merangkum seluruh paket asli di dalam bingkai VXLAN. Ini secara efektif menciptakan “paket dalam paket” dengan header baru:

```
+-----+-----+-----+-----+
+-----+
| Outer Ethernet | Outer IP      | Outer UDP    | VXLAN        | Original Pod-to-Pod
|               |               |               |              |
| Src: Node1 MAC | Src: 10.80.0.2 | Src: Random  | VNI: 42      | Packet (unchanged
|               | Dst: 10.80.0.3 | Dst: 8472    |              | from above)
|               |
+-----+-----+-----+-----+
+-----+
```

Poin-poin penting tentang enkapsulasi ini: - Paket luar dialamatkan dari Node 1 (10.80.0.2) ke Node 2 (10.80.0.3) - Port UDP 8472 adalah port VXLAN yang digunakan Cilium secara default

- VXLAN Network Identifier (VNI) mengidentifikasi jaringan overlay mana yang dimiliki paket ini - Seluruh paket asli (dengan IP Pod A sebagai sumber dan IP Pod B sebagai tujuan) dipertahankan utuh di dalam

Paket enkapsulasi sekarang memasuki tumpukan jaringan reguler Node 1 dan diproses dengan cara yang sama seperti paket lainnya:

1. Pemrosesan Jaringan Node: Tumpukan jaringan Node 1 merutekan paket berdasarkan tujuannya () 10.80.0.3
2. Pengiriman Jaringan Lokal:
 - Jika kedua node berada pada jaringan Layer 2 yang sama, paket dikirim langsung ke Node 2
 - Jika mereka berada di subnet yang berbeda, paket diteruskan ke router lokal terlebih dahulu
3. Penanganan Router: Router meneruskan paket berdasarkan tabel routingnya, mengirimkannya ke Node 2

3. Menerima Pemrosesan Node

Ketika paket yang dienkapsulasi tiba di Node 2 (): 10.80.0.3

1. Tumpukan jaringan node menerimanya dan mengidentifikasinya sebagai paket VXLAN (port UDP) 4789
2. Paket diteruskan ke antarmuka VXLAN CNI untuk diproses
4. Dekapsulasi VXLAN

CNI pada Node 2 memproses paket VXLAN:

1. Ini menghapus header luar (Ethernet, IP, UDP, dan VXLAN)
2. Ini mengekstrak paket batin asli
3. Paket sekarang kembali ke bentuk aslinya:

```
+-----+-----+-----+-----+
| Ethernet Header | IP Header      | TCP/UDP       | Payload       |
| Src: Pod A MAC  | Src: 10.85.1.56 | Src: 43721    |               |
| Dst: Gateway MAC | Dst: 10.85.2.67 | Dst: 8080     |               |
+-----+-----+-----+-----+
```

CNI pada Node 2 memeriksa IP tujuan (10.85.2.67) dan:

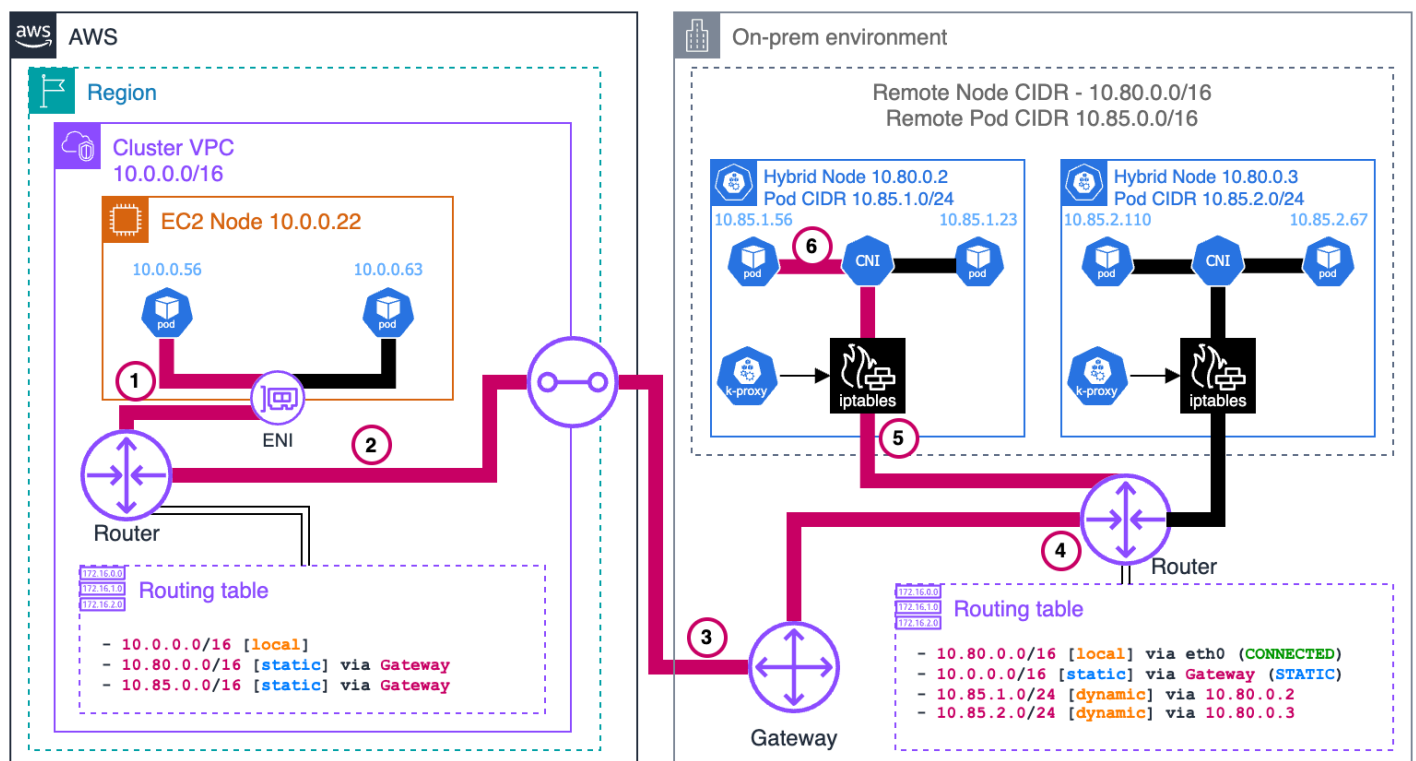
1. Mengidentifikasi bahwa IP ini milik pod lokal
2. Rutekan paket melalui antarmuka virtual yang sesuai
3. Mengirimkan paket ke namespace jaringan Pod B

Respons

Ketika Pod B merespons Pod A, seluruh proses terjadi secara terbalik:

4. Pod B mengirimkan paket ke Pod A () 10.85.1.56
5. CNI Node 2 merangkum dengan VXLAN, mengatur tujuan ke Node 1 () 10.80.0.2
6. Paket yang dienkapsulasi dikirim ke Node 1
7. CNI Node 1 mendekapsulasinya dan memberikan respons asli ke Pod A

Pod pada node cloud ke pod pada node hibrid (lalu lintas timur-barat)



Permintaan

1. Pod A Memulai Komunikasi

Pod A (10.0.0.56) pada EC2 Node ingin mengirim lalu lintas ke Pod B (10.85.1.56) pada Node Hybrid. Paket awal terlihat seperti ini:

```
+-----+-----+-----+
| IP Header | TCP Header | Payload |
| Src: 10.0.0.56 | Src: 52390 (random) | |
| Dst: 10.85.1.56 | Dst: 8080 | |
+-----+-----+-----+
```

Dengan VPC CNI, Pod A memiliki IP dari VPC CIDR dan langsung dilampirkan ke ENI pada instance. EC2 Namespace jaringan pod terhubung ke jaringan VPC, sehingga paket masuk ke infrastruktur routing VPC secara langsung.

2. Perutean VPC

Tabel rute VPC berisi rute khusus untuk Remote Pod CIDR (10.85.0.0/16), mengarahkan lalu lintas ini ke gateway: VPC-to-onprem

Destination	Target
10.0.0.0/16	local
10.85.0.0/16	vgw-id (VPC-to-onprem gateway)

Berdasarkan aturan routing ini, paket diarahkan ke gateway yang terhubung ke jaringan lokal Anda.

3. Transit Lintas Batas

Gateway mentransfer paket melintasi batas cloud melalui koneksi yang Anda buat (seperti Direct Connect atau VPN) ke jaringan lokal Anda. Paket ini mempertahankan sumber asli dan alamat IP tujuan selama transit ini.

4. Penerimaan Jaringan Lokal

Paket tiba di router lokal lokal Anda. Router berkonsultasi dengan tabel peruteannya untuk menentukan lompatan berikutnya untuk mencapai alamat 10.85.1.56. Router lokal Anda harus memiliki rute untuk pod CIDRs yang mengarahkan lalu lintas ke node hybrid yang sesuai.

Tabel router memiliki entri yang menunjukkan bahwa 10.85.1.0/24 subnet dapat dijangkau melalui node hybrid dengan IP: 10.80.0.2

Destination	Next Hop
10.85.1.0/24	10.80.0.2

5. Pemrosesan Jaringan Node

Router meneruskan paket ke node hybrid ()10.80.0.2. Ketika paket tiba di node, ia masih memiliki IP Pod A sebagai sumber dan IP Pod B sebagai tujuan.

6. Pengolahan CNI

Tumpukan jaringan node menerima paket dan, melihat bahwa IP tujuan bukan miliknya sendiri, meneruskannya ke CNI untuk diproses. CNI mengidentifikasi bahwa IP tujuan milik pod yang berjalan secara lokal pada node ini dan meneruskan paket ke pod yang benar melalui antarmuka virtual yang sesuai:

```
Original packet -> node routing -> CNI -> Pod B's network namespace
```

Pod B menerima paket dan memprosesnya sesuai kebutuhan.

Respons

6. Pod B Mengirimkan Respon

Pod B membuat paket respon dengan IP sendiri sebagai sumber dan IP Pod A sebagai tujuan:

```
+-----+-----+-----+
| IP Header | TCP Header | Payload |
| Src: 10.85.1.56 | Src: 8080 | |
| Dst: 10.0.0.56 | Dst: 52390 | |
+-----+-----+-----+
```

CNI mengidentifikasi bahwa paket ini ditujukan untuk jaringan eksternal dan meneruskannya ke tumpukan jaringan node.

5. Pemrosesan Jaringan Node

Node menentukan bahwa IP tujuan (10.0.0.56) bukan milik jaringan lokal dan meneruskan paket ke gateway defaultnya (router lokal).

4. Perutean Router Lokal

Router lokal berkonsultasi dengan tabel routing dan menentukan bahwa IP tujuan (10.0.0.56) milik VPC CIDR (). 10.0.0.0/16 Ini meneruskan paket ke gateway yang terhubung ke. AWS

3. Transit Lintas Batas

Paket berjalan kembali melalui koneksi lokal ke VPC yang sama, melintasi batas cloud dalam arah sebaliknya.

2. Perutean VPC

Ketika paket tiba di VPC, sistem routing mengidentifikasi bahwa IP tujuan milik subnet dalam VPC. Paket dirutekan melalui jaringan VPC menuju instance hosting Pod A. EC2

1. Pod A Menerima Tanggapan

Paket tiba di EC2 instance dan dikirim langsung ke Pod A melalui ENI yang terpasang. Karena VPC CNI tidak menggunakan jaringan overlay untuk pod di VPC, tidak diperlukan dekapsulasi tambahan - paket tiba dengan header aslinya utuh.

Arus lalu lintas timur-barat ini menunjukkan mengapa pod jarak jauh CIDRs harus dikonfigurasi dengan benar dan dapat dirutekan dari kedua arah:

- VPC harus memiliki rute untuk pod jarak jauh yang CIDRs mengarah ke gateway lokal
- Jaringan lokal Anda harus memiliki rute untuk pod CIDRs yang mengarahkan lalu lintas ke node tertentu yang menghosting pod tersebut.

nodeadmReferensi node hibrida

Amazon EKS Hybrid Nodes CLI (nodeadm) menyederhanakan instalasi, konfigurasi, pendaftaran, dan penghapusan instalasi komponen node hybrid. Anda dapat menyertakan nodeadm dalam gambar sistem operasi Anda untuk mengotomatiskan bootstrap node hybrid, lihat [the section called “Siapkan sistem operasi”](#) untuk informasi lebih lanjut.

nodeadmVersi untuk node hybrid berbeda dari nodeadm versi yang digunakan untuk bootstrap instance Amazon sebagai node di EC2 kluster Amazon EKS. Ikuti dokumentasi dan referensi untuk nodeadm versi yang sesuai. Halaman dokumentasi ini untuk nodeadm versi node hybrid.

Kode sumber untuk node hybrid diterbitkan di nodeadm repositori <https://github.com/aws/eks-hybrid> GitHub .

Important

Anda harus menjalankan nodeadm dengan pengguna yang memiliki hak root/sudo.

Unduh **nodeadm**

Versi node hibrida di-host di Amazon S3 yang digawangi oleh Amazon. **nodeadm** CloudFront Untuk menginstal **nodeadm** di setiap host lokal, Anda dapat menjalankan perintah berikut dari host lokal Anda.

Untuk host x86_64

```
curl -OL 'https://hybrid-assets.eks.amazonaws.com/releases/latest/bin/linux/amd64/nodeadm'
```

Untuk host ARM

```
curl -OL 'https://hybrid-assets.eks.amazonaws.com/releases/latest/bin/linux/arm64/nodeadm'
```

Tambahkan izin file yang dapat dieksekusi ke biner yang diunduh di setiap host.

```
chmod +x nodeadm
```

nodeadm install

nodeadm install Perintah ini digunakan untuk menginstal artefak dan dependensi yang diperlukan untuk menjalankan dan menggabungkan node hybrid ke cluster Amazon EKS. **nodeadm install** Perintah dapat dijalankan secara individual pada setiap node hybrid atau dapat dijalankan selama pipeline build image untuk menginstal dependensi node hybrid di image sistem operasi.

Penggunaan

```
nodeadm install [KUBERNETES_VERSION] [flags]
```

Argumen Posisi

(Wajib) **KUBERNETES_VERSION** Versi mayor.minor dari EKS Kubernetes untuk diinstal, misalnya 1.32

Bendera

Nama	Wajib	Deskripsi
<code>-p,</code> <code>--credential-provi</code> <code>der</code>	BETUL	Penyedia kredensial untuk menginstal. Nilai yang didukung adalah <code>iam-ra</code> dan <code>ssm</code> . Untuk informasi selengkapnya, lihat the section called “Siapkan kredensial” .
<code>-s,</code> <code>--containerd-source</code>	SALAH	Sumber untuk <code>containerd</code>. <code>nodeadm</code> mendukung penginstalan <code>containerd</code> dari distro OS, paket Docker, dan melewati <code>containerd</code> instalasi. Nilai-nilai <code>distro</code>- Ini adalah nilai default. <code>nodeadm</code> akan menginstal <code>containerd</code> paket didistribusikan oleh OS <code>node</code>. <code>distro</code> bukan nilai yang didukung untuk sistem operasi Red Hat Enterprise Linux (RHEL). <code>docker</code>- <code>nodeadm</code> akan menginstal <code>containerd</code> paket yang dibangun dan didistribusikan oleh Docker. <code>docker</code> bukan nilai yang didukung untuk Amazon Linux 2023 <code>none</code>- tidak <code>nodeadm</code> akan menginstal <code>containerd</code> paket. Anda harus menginsta

Nama	Wajib	Deskripsi
		l secara manual containerd sebelum menjalankan <code>nodeadm init</code> .
<code>-r,</code> <code>--region</code>	SALAH	Menentukan AWS Wilayah untuk men-download artefak seperti Agen SSM. Default ke <code>us-west-2</code> .
<code>-t,</code> <code>--timeout</code>	SALAH	Durasi perintah instalasi maksimum. Input mengikuti format durasi. Sebagai contoh, <code>1h23m</code> . Batas waktu unduhan default untuk perintah instal diatur ke 20 menit.
<code>-h, --help</code>	SALAH	Menampilkan pesan bantuan dengan parameter tanda, subperintah, dan nilai posisi yang tersedia.

Contoh

Instal versi Kubernetes dengan AWS Systems 1.32 Manager (SSM) sebagai penyedia kredensialnya

```
nodeadm install 1.32 --credential-provider ssm
```

Instal versi Kubernetes dengan AWS Systems 1.32 Manager (SSM) sebagai penyedia kredensialnya, Docker sebagai sumber containerd, dengan waktu tunggu unduhan 20 menit.

```
nodeadm install 1.32 --credential-provider ssm --containerd-source docker --timeout 20m
```

Instal versi Kubernetes 1.32 dengan AWS IAM Roles Anywhere sebagai penyedia kredensi

```
nodeadm install 1.32 --credential-provider iam-ra
```

nodeadm config check

`nodeadm config check` Perintah memeriksa konfigurasi node yang disediakan untuk kesalahan. Perintah ini dapat digunakan untuk memverifikasi dan memvalidasi kebenaran file konfigurasi node hybrid.

Penggunaan

```
nodeadm config check [flags]
```

Bendera

Nama	Wajib	Deskripsi
<code>-c,</code> <code>--config-source</code>	BETUL	Sumber konfigurasi nodeadm. Untuk node hybrid input harus mengikuti URI dengan skema file.
<code>-h, --help</code>	SALAH	Menampilkan pesan bantuan dengan parameter tanda, subperintah, dan nilai posisi yang tersedia.

Contoh

```
nodeadm config check -c file:///nodeConfig.yaml
```

nodeadm init

`nodeadm init` Perintah memulai dan menghubungkan node hybrid dengan cluster Amazon EKS yang dikonfigurasi. Lihat [the section called “Node Config untuk aktivasi hybrid SSM”](#) atau [the section called “Konfigurasi Node untuk Peran IAM Di Mana Saja”](#) untuk detail tentang cara mengkonfigurasi `nodeConfig.yaml` file.

Penggunaan

```
nodeadm init [flags]
```

Bendera

Nama	Wajib	Deskripsi
-c, --config-source	BETUL	Sumber nodeadm konfigurasi. Untuk node hybrid input harus mengikuti URI dengan skema file.
-s, --skip	SALAH	Fase yang init harus dilewati. Tidak disarankan untuk melewati salah satu fase kecuali itu membantu memperbaiki masalah. Nilai-nilai <code>install-validation</code> melewati memeriksa apakah perintah <code>install</code> sebelumnya berhasil berjalan. <code>cni-validation</code> melewati memeriksa apakah port VXLAN Cilium atau Calico CNI dibuka jika firewall diaktifkan pada node <code>node-ip-validation</code> melewati memeriksa apakah IP node termasuk dalam CIDR di jaringan node jarak jauh
-h, --help	SALAH	Menampilkan pesan bantuan dengan parameter tanda,

Nama	Wajib	Deskripsi
		subperintah, dan nilai posisi yang tersedia.

Contoh

```
nodeadm init -c file://nodeConfig.yaml
```

nodeadm upgrade

`nodeadm upgrade` Perintah memutakhirkan semua artefak yang diinstal ke versi terbaru dan bootstrap node untuk mengonfigurasi artefak yang ditingkatkan dan bergabung dengan cluster EKS. AWS Upgrade adalah perintah yang mengganggu untuk beban kerja yang berjalan pada node. Harap pindahkan beban kerja Anda ke node lain sebelum menjalankan upgrade.

Penggunaan

```
nodeadm upgrade [KUBERNETES_VERSION] [flags]
```

Argumen Posisi

(Wajib) `KUBERNETES_VERSION` Versi mayor.minor dari EKS Kubernetes untuk diinstal, misalnya 1.32

Bendera

Nama	Wajib	Deskripsi
-c, --config-source	BETUL	Sumber nodeadm konfigurasi. Untuk node hybrid input harus mengikuti URI dengan skema file.
-t, --timeout	SALAH	Batas waktu untuk mengunduh artefak. Input mengikuti format durasi. Misalnya 1h23m. Batas

Nama	Wajib	Deskripsi
		waktu unduhan default untuk perintah upgrade diatur ke 10 menit.
-s, --skip	SALAH	Fase upgrade yang akan dilewati. Tidak disarankan untuk melewati fase apapun kecuali itu membantu memperbaiki masalah. Nilai-nilai pod-validation melewati pemeriksaan apakah semua pod no berjalan pada node, kecuali set daemon dan pod statis. node-validation melewati memeriksa apakah simpul telah ditutup. init-validation melewati memeriksa apakah node telah berhasil diinisialisasi sebelum menjalankan upgrade.
-h, --help	SALAH	Menampilkan pesan bantuan dengan parameter tanda, subperintah, dan nilai posisi yang tersedia.

Contoh

```
nodeadm upgrade 1.32 -c file://nodeConfig.yaml
```

```
nodeadm upgrade 1.32 -c file://nodeConfig.yaml --timeout 20m
```

nodeadm uninstall

`nodeadm uninstall` Perintah berhenti dan menghapus `nodeadm` instalasi artefak selama `nodeadm install`, termasuk `kubelet` dan `containerd`. Catatan, perintah `uninstall` tidak menguras atau menghapus node hybrid Anda dari cluster Anda. Anda harus menjalankan operasi drain dan menghapus secara terpisah, lihat [the section called “Hapus node hybrid”](#) untuk informasi lebih lanjut. Secara default, tidak `nodeadm uninstall` akan melanjutkan jika ada pod yang tersisa di node. Demikian pula, `nodeadm uninstall` tidak menghapus dependensi atau dependensi CNI dari add-on Kubernetes lain yang Anda jalankan di kluster Anda. Untuk menghapus instalasi CNI sepenuhnya dari host Anda, lihat instruksi di [the section called “Konfigurasi CNI”](#). Jika Anda menggunakan aktivasi hibrida AWS SSM sebagai penyedia kredensi lokal, `nodeadm uninstall` perintah tersebut membatalkan pendaftaran host Anda sebagai instans yang dikelola SSM. AWS

Penggunaan

```
nodeadm uninstall [flags]
```

Bendera

Nama	Wajib	Deskripsi
<code>-s,</code> <code>--skip</code>	SALAH	Fase upgrade yang akan dilewati. Tidak disarankan untuk melewati fase apapun kecuali itu membantu memperbaiki masalah. Nilai-nilai <code>pod-validation</code> melewati pemeriksaan apakah semua pod no berjalan pada node,

Nama	Wajib	Deskripsi
		kecuali set daemon dan pod statis.
		node-validation melewati memeriksa apakah simpul telah ditutup.
		init-validation melewati memeriksa apakah node telah berhasil diinisialisasi sebelum menjalankan uninstall.
-h, --help	SALAH	Menampilkan pesan bantuan dengan parameter tanda, subperintah, dan nilai posisi yang tersedia.

Contoh

```
nodeadm uninstall
```

```
nodeadm uninstall --skip node-validation,pod-validation
```

nodeadm debug

`nodeadm debug` Perintah ini dapat digunakan untuk memecahkan masalah node hybrid yang tidak sehat atau salah konfigurasi. Ini memvalidasi persyaratan berikut ada di tempat.

- Node memiliki akses jaringan ke yang diperlukan AWS APIs untuk mendapatkan kredensial,
- Node ini bisa mendapatkan AWS kredensial untuk peran IAM Hybrid Nodes yang dikonfigurasi,
- Node memiliki akses jaringan ke titik akhir API EKS Kubernetes dan validitas sertifikat endpoint API EKS Kubernetes,

- Node dapat mengautentikasi dengan cluster EKS, identitasnya di cluster valid, dan bahwa node memiliki akses ke cluster EKS melalui VPC yang dikonfigurasi untuk cluster EKS.

Jika kesalahan ditemukan, output perintah menyarankan langkah-langkah pemecahan masalah. Langkah-langkah validasi tertentu menunjukkan proses anak. Jika ini gagal, output ditampilkan di bagian stderr di bawah kesalahan validasi.

Penggunaan

```
nodeadm debug [flags]
```

Bendera

Nama	Wajib	Deskripsi
<code>-c, --config-source</code>	BETUL	Sumber nodeadm konfigurasi. Untuk node hybrid input harus mengikuti URI dengan skema file.
<code>--no-color</code>	SALAH	Menonaktifkan output warna. Berguna untuk otomatisasi.
<code>-h, --help</code>	SALAH	Menampilkan pesan bantuan dengan parameter tanda, subperintah, dan nilai posisi yang tersedia.

Contoh

```
nodeadm debug -c file://nodeConfig.yaml
```

Lokasi file Nodeadm

instal nodeadm

Saat berjalannodeadm `install`, file dan lokasi file berikut dikonfigurasi.

Artifact	Jalur
Peran IAM Di Mana Saja CLI	/usr/local/bin/aws_penandatangan_pembantu
Biner Kubelet	/usr/bin/kubelet
Biner Kubectl	usr/local/bin/kubectl
Penyedia Kredensial ECR	/etc/eks/image-credential-provider/ecr-kredensial-penyedia
AWS Autentikator IAM	/usr/local/bin/aws-iam-autentikator
CLI Pengaturan SSM	/opt/ssm/ssm-setup-cli
SSM Agent	Di Ubuntu -/snap/amazon-ssm-agent/current/amazon-ssm-agent Pada RHEL & AL2 023 -/ssm-agent usr/bin/amazon
Kontainer	Di Ubuntu & AL2 023 -/usr/bin/containerd Pada RHEL - /bin/containerd
Iptables	Di Ubuntu & AL2 023 -/usr/sbin/iptables Pada RHEL - /sbin/iptables
Plugin CNI	/opt/cni/bin
pelacak artefak terpasang	/opt/nodeadm/tracker

nodeadm init

Saat berjalannodeadm init, file dan lokasi file berikut dikonfigurasi.

Nama	Jalur
Kubelet kubeconfig	/var/lib/kubelet/kubeconfig

Nama	Jalur
Konfigurasi Kubelet	/etc/kubernetes/kubelet/config.json
Satuan sistem Kubelet	/etc/systemd/system/kubelet.layanan
Konfigurasi penyedia kredensial gambar	/etc/eks/image-credential-provider/config.json
Berkas env Kubelet	/etc/eks/kubelet/environment
Sertifikat Kubelet	/etc/kubernetes/pki/ca.crt
Konfigurasi containerd	/etc/containerd/config.toml
Konfigurasi modul kernel containerd	/etc/modules-load.d/containerd.conf
AWS berkas konfigurasi	/etc/aws/hybrid/config
AWS berkas kredensial (jika mengaktifkan file kredensial)	/eks-hybrid/.aws/credentials
AWS penandatanganan unit sistem pembantu	/etc/systemd/system/aws_signing_helper_update.service
File conf Sysctl	/etc/sysctl.d/99-nodeadm.conf
Sertifikat CA	/etc/ssl/certs/ca-sertifikat.crt
File kunci Gpg	/etc/apt/keyrings/docker.asc
File sumber repo Docker	/etc/apt/sources.list.d/docker.daftar

Node Config untuk aktivasi hybrid SSM

Berikut ini adalah contoh `nodeConfig.yaml` saat menggunakan aktivasi hibrida AWS SSM untuk kredensial node hybrid.

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
```

```

name:           # Name of the EKS cluster
region:         # AWS Region where the EKS cluster resides
hybrid:
  ssm:
    activationCode: # SSM hybrid activation code
    activationId:   # SSM hybrid activation id

```

Konfigurasi Node untuk Peran IAM Di Mana Saja

Berikut ini adalah contoh `nodeConfig.yaml` untuk AWS IAM Roles Anywhere untuk kredensial node hybrid.

Saat menggunakan AWS IAM Roles Anywhere sebagai penyedia kredensi lokal, yang `nodeName` Anda gunakan dalam `nodeadm` konfigurasi harus selaras dengan izin yang Anda cakup untuk peran IAM Hybrid Nodes Anda. Misalnya, jika izin Anda untuk peran IAM Hybrid Nodes hanya mengizinkan AWS IAM Roles Anywhere untuk mengambil peran ketika nama sesi peran sama dengan CN sertifikat host, maka `nodeName` dalam `nodeadm` konfigurasi Anda harus sama dengan CN sertifikat Anda. `nodeName` yang Anda gunakan tidak boleh lebih dari 64 karakter. Untuk informasi selengkapnya, lihat [the section called “Siapkan kredensial”](#).

```

apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name:           # Name of the EKS cluster
    region:         # AWS Region where the EKS cluster resides
  hybrid:
    iamRolesAnywhere:
      nodeName:      # Name of the node
      trustAnchorArn: # ARN of the IAM Roles Anywhere trust anchor
      profileArn:     # ARN of the IAM Roles Anywhere profile
      roleArn:        # ARN of the Hybrid Nodes IAM role
      certificatePath: # Path to the certificate file to authenticate with the IAM
Roles Anywhere trust anchor
      privateKeyPath: # Path to the private key file for the certificate

```

Node Config untuk menyesuaikan kubelet (Opsional)

Anda dapat meneruskan konfigurasi dan flag kubelet dalam konfigurasi Anda. `nodeadm` Lihat contoh di bawah ini untuk cara menambahkan label node tambahan `abc.amazonaws.com/test-label` dan konfigurasi untuk pengaturan `shutdownGracePeriod` ke 30 detik.

```

apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name:          # Name of the EKS cluster
    region:        # AWS Region where the EKS cluster resides
  kubelet:
    config:        # Map of kubelet config and values
      shutdownGracePeriod: 30s
    flags:         # List of kubelet flags
      - --node-labels=abc.company.com/test-label=true
  hybrid:
    ssm:
      activationCode: # SSM hybrid activation code
      activationId:   # SSM hybrid activation id

```

Node Config untuk menyesuaikan containerd (Opsional)

Anda dapat meneruskan konfigurasi containerd kustom dalam konfigurasi Anda. nodeadm Konfigurasi containerd nodeadm untuk menerima TOML in-line. Lihat contoh di bawah ini untuk cara mengonfigurasi containerd untuk menonaktifkan penghapusan layer gambar yang tidak dikemas di toko konten containerd.

```

apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name:          # Name of the EKS cluster
    region:        # AWS Region where the EKS cluster resides
  containerd:
    config: |      # Inline TOML containerd additional configuration
      [plugins."io.containerd.grpc.v1.cri".containerd]
      discard_unpacked_layers = false
  hybrid:
    ssm:
      activationCode: # SSM hybrid activation code
      activationId:   # SSM hybrid activation id

```

Anda juga dapat menggunakan konfigurasi containerd untuk mengaktifkan dukungan. SELinux Dengan SELinux diaktifkan pada containerd, pastikan pod yang dijadwalkan pada node memiliki

SecurityContext yang tepat dan diaktifkan. `seLinuxOptions` Informasi lebih lanjut tentang konfigurasi konteks keamanan dapat ditemukan di dokumentasi [Kubernetes](#).

Note

Red Hat Enterprise Linux (RHEL) 8 dan RHEL 9 telah SELinux diaktifkan secara default dan diatur ke ketat pada host. Amazon Linux 2023 telah SELinux diaktifkan secara default dan disetel ke mode permisif. Kapan SELinux diatur ke mode permisif pada host, mengaktifkannya di containerd tidak akan memblokir permintaan tetapi akan mencatatnya sesuai dengan konfigurasi pada host. SELinux

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name:           # Name of the EKS cluster
    region:         # AWS Region where the EKS cluster resides
  containerd:
    config: |        # Inline TOML containerd additional configuration
      [plugins."io.containerd.grpc.v1.cri"]
      enable_selinux = true
  hybrid:
    ssm:
      activationCode: # SSM hybrid activation code
      activationId:   # SSM hybrid activation id
```

Memecahkan masalah node hybrid

Topik ini mencakup beberapa kesalahan umum yang mungkin Anda lihat saat menggunakan Amazon EKS Hybrid Nodes dan cara memperbaikinya. Untuk informasi pemecahan masalah lainnya, lihat [Pemecahan Masalah](#) dan [tag Pusat Pengetahuan untuk Amazon EKS](#) di AWS re:Post. Jika Anda tidak dapat menyelesaikan masalah, hubungi AWS Support.

Pemecahan masalah node dengan **nodeadm debug** Anda dapat menjalankan `nodeadm debug` perintah dari node hybrid Anda untuk memvalidasi jaringan dan persyaratan kredensi terpenuhi. Untuk informasi lebih lanjut tentang `nodeadm debug` perintah, lihat [the section called “Node hibrida nodeadm”](#).

Mendeteksi masalah dengan node hibrid Anda dengan wawasan kluster Amazon EKS cluster insights mencakup pemeriksaan wawasan yang mendeteksi masalah umum dengan konfigurasi EKS Hybrid Nodes di kluster Anda. Anda dapat melihat hasil semua pemeriksaan wawasan dari AWS Management Console, AWS CLI, dan AWS SDKs Untuk informasi selengkapnya tentang wawasan cluster, lihat [the section called “Wawasan cluster”](#).

Menginstal pemecahan masalah node hybrid

Topik pemecahan masalah berikut terkait dengan menginstal dependensi node hybrid pada host dengan perintah. `nodeadm install`

nodeadm perintah gagal **must run as root**

`nodeadm install` Perintah harus dijalankan dengan pengguna yang memiliki root atau sudo hak istimewa pada host Anda. Jika Anda menjalankan `nodeadm install` dengan pengguna yang tidak memiliki root atau sudo hak istimewa, Anda akan melihat kesalahan berikut dalam `nodeadm` output.

```
"msg":"Command failed","error":"must run as root"
```

Tidak dapat terhubung ke dependensi

`nodeadm install` Perintah menginstal dependensi yang diperlukan untuk node hybrid. Dependensi node hybrid mencakup `containerd`, `kubelet`, `kubectl`, dan komponen AWS SSM atau AWS IAM Roles Anywhere. Anda harus memiliki akses dari tempat Anda menjalankan `nodeadm install` untuk mengunduh dependensi ini. Untuk informasi lebih lanjut tentang daftar lokasi yang harus dapat Anda akses, lihat [the section called “Siapkan jaringan”](#). Jika Anda tidak memiliki akses, Anda akan melihat kesalahan yang mirip dengan yang berikut dalam `nodeadm install` output.

```
"msg":"Command failed","error":"failed reading file from url: ...: max retries achieved for http request"
```

Gagal memperbarui pengelola paket

`nodeadm install` Perintah berjalan `apt update` atau `yum update` atau `dnf update` sebelum menginstal dependensi node hybrid. Jika langkah ini tidak berhasil, Anda mungkin melihat kesalahan yang mirip dengan berikut ini. Untuk memulihkan, Anda dapat menjalankan `apt update` atau `yum update` atau `dnf update` sebelum berlari `nodeadm install` atau Anda dapat mencoba menjalankan kembali `nodeadm install`.

```
failed to run update using package manager
```

Batas waktu tunggu atau batas waktu konteks terlampaui

Saat menjalankannodeadm install, jika Anda melihat masalah pada berbagai tahap proses penginstalan dengan kesalahan yang menunjukkan ada batas waktu atau batas waktu konteks terlampaui, Anda mungkin memiliki koneksi lambat yang mencegah penginstalan dependensi node hibrida sebelum batas waktu terpenuhi. Untuk mengatasi masalah ini, Anda dapat mencoba menggunakan `--timeout` tanda nodeadm untuk memperpanjang durasi batas waktu untuk mengunduh dependensi.

```
nodeadm install K8S_VERSION --credential-provider CRED_PROVIDER --timeout 20m0s
```

Menghubungkan pemecahan masalah node hybrid

Topik pemecahan masalah di bagian ini terkait dengan proses menghubungkan node hybrid ke cluster EKS dengan perintah `nodeadm init`

Kesalahan operasi atau skema yang tidak didukung

Saat menjalankannodeadm init, jika Anda melihat kesalahan yang terkait dengan `operation error` atau `unsupported scheme`, periksa kesalahan Anda `nodeConfig.yaml` untuk memastikannya diformat dengan benar dan diteruskan kenodeadm. Untuk informasi selengkapnya tentang format dan opsi untuk `nodeConfig.yaml`, lihat [the section called “Node hibrida nodeadm”](#).

```
"msg":"Command failed","error":"operation error ec2imds: GetRegion, request canceled, context deadline exceeded"
```

Peran IAM Node Hybrid tidak memiliki izin untuk tindakan **eks:DescribeCluster**

Saat menjalankannodeadm init, cobalah nodeadm mengumpulkan informasi tentang klaster EKS Anda dengan memanggil Describe Cluster. Jika peran IAM Hybrid Nodes Anda tidak memiliki izin untuk `eks:DescribeCluster` tindakan tersebut. Untuk informasi selengkapnya tentang izin yang diperlukan untuk peran IAM Hybrid Nodes, lihat [the section called “Siapkan kredensial”](#)

```
"msg":"Command failed","error":"operation error EKS: DescribeCluster, https response error StatusCode: 403 ... AccessDeniedException"
```

Node IP tidak di jaringan simpul jarak jauh CIDR

Saat menjalankannodeadm init, Anda mungkin mengalami kesalahan jika alamat IP node tidak berada dalam jaringan node jarak jauh yang ditentukan CIDRs. Kesalahan akan terlihat mirip dengan contoh berikut:

```
node IP 10.18.0.1 is not in any of the remote network CIDR blocks [10.0.0.0/16
192.168.0.0/16]
```

Contoh ini menunjukkan node dengan IP 10.18.0.1 mencoba bergabung dengan cluster dengan jaringan jarak jauh 10.0.0.0/16 dan CIDRs 192.168.0.0/16. Kesalahan terjadi karena 10.18.0.1 tidak berada dalam salah satu rentang.

Konfirmasikan bahwa Anda telah mengonfigurasi dengan benar RemoteNodeNetworks untuk menyertakan semua alamat IP node. Untuk informasi selengkapnya tentang konfigurasi jaringan, lihat[the section called “Siapkan jaringan”](#).

- Jalankan perintah berikut di wilayah klaster Anda berada untuk memeriksa RemoteNodeNetwork konfigurasi Anda. Verifikasi bahwa blok CIDR yang tercantum dalam output menyertakan rentang IP node Anda dan sama dengan blok CIDR yang tercantum dalam pesan kesalahan. Jika tidak cocok, konfirmasikan nama klaster dan wilayah yang nodeConfig.yaml sesuai dengan klaster yang Anda inginkan.

```
aws eks describe-cluster --name CLUSTER_NAME --region REGION_NAME --query
cluster.remoteNetworkConfig.remoteNodeNetworks
```

- Verifikasi bahwa Anda bekerja dengan node yang dimaksud:
 - Konfirmasikan bahwa Anda berada di node yang benar dengan memeriksa nama host dan alamat IPnya cocok dengan yang ingin Anda daftarkan ke cluster.
 - Konfirmasikan bahwa node ini berada di jaringan lokal yang benar (yang jangkauan CIDR-nya terdaftar RemoteNodeNetwork selama penyiapan cluster).

Jika IP node Anda masih tidak seperti yang Anda harapkan, periksa hal berikut:

- Jika Anda menggunakan IAM Roles Anywhere, kubelet lakukan pencarian DNS pada IAM Roles Anywhere nodeName dan menggunakan IP yang terkait dengan nama node jika tersedia. Jika Anda mempertahankan entri DNS untuk node Anda, konfirmasikan bahwa entri ini mengarah ke IPs dalam jaringan node jarak jauh Anda. CIDRs

- Jika node Anda memiliki beberapa antarmuka jaringan, kubelet mungkin pilih antarmuka dengan alamat IP di luar jaringan node jarak jauh Anda CIDRs sebagai default. Untuk menggunakan antarmuka yang berbeda, tentukan alamat IP-nya menggunakan `--node-ip` kubelet bendera di `nodeConfig.yaml`. Untuk informasi selengkapnya, lihat [the section called “Node hibrida nodeadm”](#). Anda dapat melihat antarmuka jaringan node Anda dan alamat IP-nya dengan menjalankan perintah berikut pada node Anda:

```
ip addr show
```

Node hibrida tidak muncul di kluster EKS

Jika Anda menjalankan `nodeadm init` dan itu selesai tetapi node hybrid Anda tidak muncul di kluster Anda, mungkin ada masalah dengan koneksi jaringan antara node hybrid Anda dan bidang kontrol EKS, Anda mungkin tidak memiliki izin grup keamanan yang diperlukan yang dikonfigurasi, atau Anda mungkin tidak memiliki pemetaan yang diperlukan dari peran IAM Hybrid Nodes Anda ke Kubernetes Role-Based Access Control (RBAC). Anda dapat memulai proses debugging dengan memeriksa status kubelet dan kubelet log dengan perintah berikut. Jalankan perintah berikut dari node hybrid yang gagal bergabung dengan cluster Anda.

```
systemctl status kubelet
```

```
journalctl -u kubelet -f
```

Tidak dapat berkomunikasi dengan cluster

Jika node hybrid Anda tidak dapat berkomunikasi dengan bidang kontrol cluster, Anda mungkin melihat log yang mirip dengan berikut ini.

```
"Failed to ensure lease exists, will retry" err="Get ..."
```

```
"Unable to register node with API server" err="Post ..."
```

```
Failed to contact API server when waiting for CSINode publishing ... dial tcp <ip address>: i/o timeout
```

Jika Anda melihat pesan ini, periksa hal berikut untuk memastikannya memenuhi persyaratan node hibrida yang dirinci [the section called “Siapkan jaringan”](#).

- Konfirmasikan bahwa VPC yang diteruskan ke kluster EKS memiliki rute ke Transit Gateway (TGW) atau Virtual Private Gateway (VGW) untuk node lokal dan pod opsional Anda. CIDRs
- Konfirmasikan bahwa Anda memiliki grup keamanan tambahan untuk kluster EKS Anda yang memiliki aturan masuk untuk node lokal CIDRs dan pod opsional. CIDRs
- Konfirmasikan router lokal Anda dikonfigurasi untuk mengizinkan lalu lintas ke dan dari bidang kontrol EKS.

Tidak sah

Jika node hibrida Anda dapat berkomunikasi dengan bidang kontrol EKS tetapi tidak dapat mendaftar, Anda mungkin melihat log yang mirip dengan berikut ini. Perhatikan perbedaan utama dalam pesan log di bawah ini adalah Unauthorized kesalahannya. Ini menandakan bahwa node tidak dapat melakukan tugasnya karena tidak memiliki izin yang diperlukan.

```
"Failed to ensure lease exists, will retry" err="Unauthorized"
```

```
"Unable to register node with API server" err="Unauthorized"
```

```
Failed to contact API server when waiting for CSINode publishing: Unauthorized
```

Jika Anda melihat pesan ini, periksa hal berikut untuk memastikannya memenuhi detail persyaratan node hibrida di [the section called “Siapkan kredensial”](#) dan [the section called “Siapkan akses cluster”](#).

- Konfirmasikan identitas node hybrid sesuai dengan peran IAM Hybrid Nodes yang Anda harapkan. Ini dapat dilakukan dengan menjalankan `sudo aws sts get-caller-identity` dari node hybrid Anda.
- Konfirmasikan peran IAM Hybrid Nodes Anda memiliki izin yang diperlukan.
- Konfirmasikan bahwa di kluster Anda, Anda memiliki entri akses EKS untuk peran IAM Hybrid Nodes Anda atau konfirmasikan bahwa Anda `aws-auth ConfigMap` memiliki entri untuk peran IAM Hybrid Nodes Anda. Jika Anda menggunakan entri akses EKS, konfirmasikan entri akses Anda untuk peran IAM Node Hybrid Anda memiliki jenis `HYBRID_LINUX` akses. Jika Anda menggunakan `aws-auth ConfigMap`, konfirmasikan entri Anda untuk peran IAM Hybrid Nodes memenuhi persyaratan dan pemformatan yang dirinci. [the section called “Siapkan akses cluster”](#)

Node hibrida terdaftar dengan kluster EKS tetapi menunjukkan status **Not Ready**

Jika node hibrida Anda berhasil terdaftar dengan kluster EKS Anda, tetapi node hibrida menunjukkan status `Not Ready`, hal pertama yang harus diperiksa adalah status Container Networking Interface (CNI) Anda. Jika Anda belum menginstal CNI, maka diharapkan node hybrid Anda memiliki status `Not Ready`. Setelah CNI diinstal dan berjalan dengan sukses, node diperbarui ke status `Ready`. Jika Anda mencoba menginstal CNI tetapi tidak berhasil berjalan, lihat [the section called “Pemecahan masalah CNI node hibrida”](#) di halaman ini.

Permintaan Penandatanganan Sertifikat (CSRs) macet Tertunda

Setelah menghubungkan node hybrid ke cluster EKS Anda, jika Anda melihat ada node hybrid yang tertunda CSRs, node hybrid Anda tidak memenuhi persyaratan untuk persetujuan otomatis. CSRs untuk node hybrid secara otomatis disetujui dan dikeluarkan jika node CSRs for hybrid dibuat oleh node dengan `eks.amazonaws.com/compute-type: hybrid` label, dan CSR memiliki Nama Alternatif Subjek berikut (SANs): setidaknya satu DNS SAN sama dengan nama node dan IP SANs milik jaringan node jarak jauh. CIDRs

Profil hybrid sudah ada

Jika Anda mengubah `nodeadm` konfigurasi dan mencoba mendaftarkan ulang node dengan konfigurasi baru, Anda mungkin melihat kesalahan yang menyatakan bahwa profil hibrida sudah ada tetapi isinya telah berubah. Alih-alih berjalan `nodeadm init` di antara perubahan konfigurasi, jalankan `nodeadm uninstall` diikuti oleh `nodeadm install` dan `nodeadm init`. Ini memastikan pembersihan yang tepat dengan perubahan konfigurasi.

```
"msg":"Command failed","error":"hybrid profile already exists at /etc/aws/hybrid/config but its contents do not align with the expected configuration"
```

Node hibrida gagal menyelesaikan API Pribadi

Setelah berjalan `nodeadm init`, jika Anda melihat kesalahan pada `kubelet` log yang menunjukkan kegagalan untuk menghubungi server API EKS Kubernetes karena `no such host`, Anda mungkin harus mengubah entri DNS untuk titik akhir API EKS Kubernetes di jaringan lokal atau di tingkat host. Lihat [Meneruskan kueri DNS masuk ke VPC Anda](#) di dokumentasi Route53. AWS

```
Failed to contact API server when waiting for CSINode publishing: Get ... no such host
```

Tidak dapat melihat node hybrid di konsol EKS

Jika Anda telah mendaftarkan node hybrid tetapi tidak dapat melihatnya di konsol EKS, periksa izin utama IAM yang Anda gunakan untuk melihat konsol. Prinsipal IAM yang Anda gunakan harus memiliki izin IAM dan Kubernetes minimum tertentu untuk melihat sumber daya di konsol. Untuk informasi selengkapnya, lihat [the section called “Akses sumber daya klaster”](#).

Menjalankan pemecahan masalah node hybrid

Jika node hibrida Anda terdaftar dengan kluster EKS AndaReady, memiliki status, dan kemudian beralih ke statusNot Ready, ada berbagai masalah yang mungkin berkontribusi pada status tidak sehat seperti node yang kekurangan sumber daya yang cukup untuk CPU, memori, atau ruang disk yang tersedia, atau node terputus dari bidang kontrol EKS. Anda dapat menggunakan langkah-langkah di bawah ini untuk memecahkan masalah node Anda, dan jika Anda tidak dapat menyelesaikan masalah, hubungi Support AWS .

Jalankan `nodeadm debug` dari node hybrid Anda untuk memvalidasi jaringan dan persyaratan kredensi terpenuhi. Untuk informasi lebih lanjut tentang `nodeadm debug` perintah, lihat [the section called “Node hibrida nodeadm”](#).

Dapatkan status node

```
kubectl get nodes -o wide
```

Periksa kondisi dan peristiwa node

```
kubectl describe node NODE_NAME
```

Dapatkan status pod

```
kubectl get pods -A -o wide
```

Periksa kondisi dan acara pod

```
kubectl describe pod POD_NAME
```

Periksa log pod

```
kubectl logs POD_NAME
```

Periksa **kubectl** log

```
systemctl status kubelet
```

```
journalctl -u kubelet -f
```

Probe keaktifan pod gagal atau webhook tidak berfungsi

Jika aplikasi, add-on, atau webhook yang berjalan pada node hybrid Anda tidak dimulai dengan benar, Anda mungkin memiliki masalah jaringan yang memblokir komunikasi ke pod. Agar control plane EKS dapat menghubungi webhook yang berjalan pada node hybrid, Anda harus mengonfigurasi kluster EKS Anda dengan jaringan pod jarak jauh dan memiliki rute untuk pod CIDR lokal di tabel perutean VPC dengan target sebagai Transit Gateway (TGW), virtual private gateway (VPW), atau gateway lain yang Anda gunakan untuk menghubungkan VPC dengan jaringan lokal Anda. Untuk informasi lebih lanjut tentang persyaratan jaringan untuk node hybrid, lihat [the section called “Siapkan jaringan”](#). Anda juga harus mengizinkan lalu lintas ini di firewall lokal Anda dan memastikan router Anda dapat merutekan dengan benar ke pod Anda. Lihat [the section called “Konfigurasi webhook”](#) untuk informasi selengkapnya tentang persyaratan untuk menjalankan webhook pada node hybrid.

Pesan log pod umum untuk skenario ini ditampilkan di bawah ini dimana ip-address adalah IP Cluster untuk layanan Kubernetes.

```
dial tcp <ip-address>:443: connect: no route to host
```

kubectl logs atau **kubectl exec** perintah tidak berfungsi

Jika **kubectl logs** atau **kubectl exec** perintah habis sementara **kubectl** perintah lain berhasil, masalah ini kemungkinan terkait dengan konfigurasi jaringan jarak jauh. Perintah-perintah ini terhubung melalui cluster ke kubelet titik akhir pada node. Untuk mengetahui informasi selengkapnya, lihat [the section called “Titik akhir kubelet”](#).

Verifikasi bahwa node IPs dan pod Anda IPs termasuk dalam jaringan node jarak jauh dan jaringan pod jarak jauh yang CIDRs dikonfigurasi untuk kluster Anda. Gunakan perintah di bawah ini untuk memeriksa tugas IP.

```
kubectl get nodes -o wide
```

```
kubectl get pods -A -o wide
```

Bandingkan ini IPs dengan jaringan jarak jauh yang dikonfigurasi CIDRs untuk memastikan perutean yang tepat. Untuk persyaratan konfigurasi jaringan, lihat [the section called “Siapkan jaringan”](#).

Pemecahan masalah CNI node hibrida

Jika Anda mengalami masalah dengan awalnya memulai Cilium atau Calico dengan node hibrida, ini paling sering disebabkan oleh masalah jaringan antara node hibrida atau pod CNI yang berjalan pada node hibrida, dan bidang kontrol EKS. Pastikan lingkungan Anda memenuhi persyaratan dalam Mempersiapkan jaringan untuk node hybrid. Ini berguna untuk memecah masalah menjadi beberapa bagian.

Konfigurasi kluster EKS

Apakah konfigurasi RemoteNodeNetwork dan RemotePodNetwork konfigurasi benar?

Konfigurasi VPC

Apakah ada rute untuk RemoteNodeNetwork dan RemotePodNetwork di tabel perutean VPC dengan target Transit Gateway atau Virtual Private Gateway?

Konfigurasi grup keamanan

Apakah ada aturan masuk dan keluar untuk dan? RemoteNodeNetwork RemotePodNetwork

Jaringan lokal

Apakah ada rute dan akses ke dan dari bidang kontrol EKS dan ke dan dari node hibrida dan pod yang berjalan pada node hybrid?

Konfigurasi CNI

Jika menggunakan jaringan overlay, apakah konfigurasi kumpulan IP untuk CNI cocok dengan yang RemotePodNetwork dikonfigurasi untuk cluster EKS jika menggunakan webhook?

Hybrid node memiliki status **Ready** tanpa CNI diinstal

Jika node hybrid Anda menunjukkan statusReady, tetapi Anda belum menginstal CNI di cluster Anda, ada kemungkinan bahwa ada artefak CNI lama pada node hybrid Anda. Secara default, ketika Anda menghapus Cilium dan Calico dengan alat seperti Helm, sumber daya pada disk tidak dihapus dari mesin fisik atau virtual Anda. Selain itu, Definisi Sumber Daya Kustom (CRDs) untuk ini CNIs

mungkin masih ada di cluster Anda dari instalasi lama. Untuk informasi selengkapnya, lihat bagian Hapus Cilium dan Hapus Calico dari [the section called “Konfigurasi CNI”](#)

Pemecahan masalah Cilium

Jika Anda mengalami masalah saat menjalankan Cilium pada node hybrid, lihat [langkah pemecahan masalah](#) dalam dokumentasi Cilium. Bagian di bawah ini mencakup masalah yang mungkin unik untuk menerapkan Cilium pada node hibrida.

Cilium tidak dimulai

Jika agen Cilium yang berjalan pada setiap node hibrida tidak dimulai, periksa log pod agen Cilium untuk mengetahui kesalahan. Agen Cilium memerlukan konektivitas ke titik akhir API EKS Kubernetes untuk memulai. Startup agen cilium akan gagal jika konektivitas ini tidak dikonfigurasi dengan benar. Dalam hal ini, Anda akan melihat pesan log yang mirip dengan yang berikut ini di log pod agen Cilium.

```
msg="Unable to contact k8s api-server"
level=fatal msg="failed to start: Get \"https://<k8s-cluster-ip>:443/api/v1/namespaces/
kube-system\": dial tcp <k8s-cluster-ip>:443: i/o timeout"
```

Agen Cilium berjalan di jaringan host. Kluster EKS Anda harus dikonfigurasi RemoteNodeNetwork untuk konektivitas Cilium. Konfirmasikan bahwa Anda memiliki grup keamanan tambahan untuk kluster EKS Anda dengan aturan masuk untuk AndaRemoteNodeNetwork, bahwa Anda memiliki rute di VPC untuk RemoteNodeNetwork Anda, dan bahwa jaringan lokal Anda dikonfigurasi dengan benar untuk memungkinkan konektivitas ke bidang kontrol EKS.

Jika operator Cilium sedang berjalan dan beberapa agen Cilium Anda berjalan tetapi tidak semuanya, konfirmasikan bahwa Anda memiliki pod yang tersedia IPs untuk dialokasikan untuk semua node di kluster Anda. Anda mengonfigurasi ukuran Pod yang dapat dialokasikan CIDRs saat menggunakan kumpulan kluster IPAM dengan `clusterPoolIPv4PodCIDRList` konfigurasi Cilium Anda. Ukuran CIDR per-node dikonfigurasi dengan `clusterPoolIPv4MaskSize` pengaturan dalam konfigurasi Cilium Anda. Lihat [Memperluas kumpulan cluster](#) di dokumentasi Cilium untuk informasi selengkapnya.

Cilium BGP tidak berfungsi

Jika Anda menggunakan Cilium BGP Control Plane untuk mengiklankan pod atau alamat layanan ke jaringan lokal, Anda dapat menggunakan perintah Cilium CLI berikut untuk memeriksa apakah

BGP mengiklankan rute ke sumber daya Anda. Untuk langkah-langkah menginstal CLI Cilium, lihat [Menginstal CLI Cilium di dokumentasi Cilium](#).

Jika BGP bekerja dengan benar, Anda harus node hybrid Anda dengan Session State established di output. Anda mungkin perlu bekerja dengan tim jaringan Anda untuk mengidentifikasi nilai yang benar untuk Local AS, Peer AS, dan Peer Address lingkungan Anda.

```
cilium bgp peers
```

```
cilium bgp routes
```

Jika Anda menggunakan Cilium BGP untuk mengiklankan Layanan dengan tipeLoadBalancer, Anda harus memiliki label yang sama pada Layanan Anda CiliumLoadBalancerIPPool dan Layanan, yang harus digunakan dalam pemilih Anda. IPs CiliumBGPAdvertisement Contoh ditunjukkan di bawah ini. Catatan, jika Anda menggunakan Cilium BGP untuk mengiklankan Layanan dengan tipe LoadBalancer, rute BGP mungkin terganggu selama agen Cilium memulai kembali. IPs Untuk informasi selengkapnya, lihat [Skenario Kegagalan](#) dalam dokumentasi Cilium.

Layanan

```
kind: Service
apiVersion: v1
metadata:
  name: guestbook
  labels:
    app: guestbook
spec:
  ports:
    - port: 3000
      targetPort: http-server
  selector:
    app: guestbook
  type: LoadBalancer
```

CiliumLoadBalancerIPPool

```
apiVersion: cilium.io/v2alpha1
kind: CiliumLoadBalancerIPPool
metadata:
  name: guestbook-pool
```



```

labels:
  app: guestbook
spec:
  blocks:
    - cidr: <CIDR to advertise>
  serviceSelector:
    matchExpressions:
      - { key: app, operator: In, values: [ guestbook ] }

```

silia BGPAdvertisement

```

apiVersion: cilium.io/v2alpha1
kind: CiliumBGPAdvertisement
metadata:
  name: bgp-advertisements-guestbook
  labels:
    advertise: bgp
spec:
  advertisements:
    - advertisementType: "Service"
      service:
        addresses:
          - ExternalIP
          - LoadBalancerIP
      selector:
        matchExpressions:
          - { key: app, operator: In, values: [ guestbook ] }

```

Pemecahan masalah Calico

Jika Anda mengalami masalah saat menjalankan Calico pada node hybrid, lihat [langkah pemecahan masalah](#) dalam dokumentasi Calico. Bagian di bawah ini mencakup masalah yang mungkin unik untuk menerapkan Calico pada node hybrid.

Tabel di bawah ini merangkum komponen Calico dan apakah mereka berjalan pada node atau jaringan pod secara default. Jika Anda mengonfigurasi Calico untuk menggunakan NAT untuk lalu lintas pod keluar, jaringan lokal Anda harus dikonfigurasi untuk merutekan lalu lintas ke CIDR node lokal dan tabel perutean VPC Anda harus dikonfigurasi dengan rute untuk CIDR node lokal dengan gateway transit (TGW) atau virtual private gateway (VGW) sebagai target. Jika Anda tidak mengonfigurasi Calico untuk menggunakan NAT untuk lalu lintas pod keluar, jaringan lokal Anda harus dikonfigurasi untuk merutekan lalu lintas ke CIDR pod lokal dan tabel perutean VPC Anda

harus dikonfigurasi dengan rute untuk CIDR pod lokal dengan gateway transit (TGW) atau virtual private gateway (VGW) sebagai target.

Komponen	Jaringan
Peladen API Calico	Simpul
Pengontrol Calico untuk Kubernetes	Pod
Agen simpul Calico	Simpul
Calico typha	Simpul
Driver simpul Calico CSI	Pod
Operator Calico	Simpul

Sumber daya Calico dijadwalkan atau berjalan pada node yang ditutup

Sumber daya Calico yang tidak berjalan sebagai a DaemonSet memiliki toleransi fleksibel secara default yang memungkinkannya dijadwalkan pada node yang dikeping yang tidak siap untuk penjadwalan atau menjalankan pod. Anda dapat memperketat toleransi untuk sumber daya DaemonSet non-Calico dengan mengubah instalasi operator Anda untuk menyertakan yang berikut ini.

```

installation:
  ...
  controlPlaneTolerations:
    - effect: NoExecute
      key: node.kubernetes.io/unreachable
      operator: Exists
      tolerationSeconds: 300
    - effect: NoExecute
      key: node.kubernetes.io/not-ready
      operator: Exists
      tolerationSeconds: 300
  calicoKubeControllersDeployment:
    spec:
      template:
        spec:
          tolerations:

```

```

      - effect: NoExecute
        key: node.kubernetes.io/unreachable
        operator: Exists
        tolerationSeconds: 300
      - effect: NoExecute
        key: node.kubernetes.io/not-ready
        operator: Exists
        tolerationSeconds: 300
    typhaDeployment:
      spec:
        template:
          spec:
            tolerations:
              - effect: NoExecute
                key: node.kubernetes.io/unreachable
                operator: Exists
                tolerationSeconds: 300
              - effect: NoExecute
                key: node.kubernetes.io/not-ready
                operator: Exists
                tolerationSeconds: 300

```

Pemecahan masalah kredensil

Untuk aktivasi hybrid AWS SSM dan Peran AWS IAM Anywhere, Anda dapat memvalidasi bahwa kredensil untuk peran IAM Hybrid Nodes dikonfigurasi dengan benar pada node hybrid Anda dengan menjalankan perintah berikut dari node hybrid Anda. Konfirmasikan nama node dan nama Peran IAM Node Hybrid adalah apa yang Anda harapkan.

```
sudo aws sts get-caller-identity
```

```

{
  "UserId": "ABCDEFGHJKLM12345678910:<node-name>",
  "Account": "<aws-account-id>",
  "Arn": "arn:aws: sts::<aws-account-id>:assumed-role/<hybrid-nodes-iam-role/<node-
name>"
}

```

AWS Pemecahan masalah Systems Manager (SSM)

Jika Anda menggunakan aktivasi hibrida AWS SSM untuk kredensil node hybrid Anda, perhatikan direktori dan artefak SSM berikut yang diinstal pada node hybrid Anda oleh. nodeadm Untuk

informasi selengkapnya tentang agen SSM, lihat [Bekerja dengan agen SSM](#) di Panduan Pengguna AWS Systems Manager.

Deskripsi	Lokasi
Agen SSM	Ubuntu - /snap/amazon-ssm-agent/ current/amazon-ssm-agent RHEL & AL2 023 - /usr/bin/amazon-ssm-agent
Log SSM Agent	/var/log/amazon/ssm
AWS kredensialnya	/root/.aws/credentials
CLI Pengaturan SSM	/opt/ssm/ssm-setup-cli

Memulai ulang agen SSM

Beberapa masalah dapat diselesaikan dengan memulai kembali agen SSM. Anda dapat menggunakan perintah di bawah ini untuk memulai ulang.

AL2023 dan sistem operasi lainnya

```
systemctl restart amazon-ssm-agent
```

Ubuntu

```
systemctl restart snap.amazon-ssm-agent.amazon-ssm-agent
```

Periksa konektivitas ke titik akhir SSM

Konfirmasikan bahwa Anda dapat terhubung ke titik akhir SSM dari node hybrid Anda. Untuk daftar titik akhir SSM, lihat titik akhir dan kuota [AWS Systems Manager](#). Ganti perintah us-west-2 di bawah ini dengan AWS Region untuk aktivasi hybrid AWS SSM Anda.

```
ping ssm.us-west-2.amazonaws.com
```

Lihat status koneksi instans SSM terdaftar

Anda dapat memeriksa status koneksi dari instance yang terdaftar dengan aktivasi hibrida SSM dengan perintah CLI AWS berikut. Ganti ID mesin dengan ID mesin instance Anda.

```
aws ssm get-connection-status --target mi-012345678abcdefgh
```

Ketidakcocokan checksum CLI Pengaturan SSM

Saat menjalankan `nodeadm install` jika Anda melihat masalah dengan ketidakcocokan `ssm-setup-cli` checksum, Anda harus mengonfirmasi bahwa tidak ada instalasi SSM yang lebih lama di host Anda. Jika ada instalasi SSM yang lebih lama di host Anda, hapus dan jalankan kembali `nodeadm install` untuk menyelesaikan masalah.

```
Failed to perform agent-installation/on-prem registration: error while verifying
installed ssm-setup-cli checksum: checksum mismatch with latest ssm-setup-cli.
```

SSM **InvalidActivation**

Jika Anda melihat kesalahan saat mendaftarkan instans Anda dengan AWS SSM, konfirmasikan `regionactivationCode`, dan `activationId` di dalamnya benar. `nodeConfig.yaml` AWS Wilayah untuk kluster EKS Anda harus sesuai dengan wilayah aktivasi hibrida SSM Anda. Jika nilai-nilai ini salah dikonfigurasi, Anda mungkin melihat kesalahan yang mirip dengan berikut ini.

```
ERROR Registration failed due to error registering the instance with AWS SSM.
InvalidActivation
```

SSMExpiredTokenException: Token keamanan yang termasuk dalam permintaan sudah kedaluwarsa

Jika agen SSM tidak dapat menyegarkan kredensial, Anda mungkin melihat file.

ExpiredTokenException Dalam skenario ini, jika Anda dapat terhubung ke titik akhir SSM dari node hybrid Anda, Anda mungkin perlu me-restart agen SSM untuk memaksa penyegaran kredensial.

```
"msg":"Command failed","error":"operation error SSM: DescribeInstanceInformation, https
response error StatusCode: 400, RequestID: eee03a9e-f7cc-470a-9647-73d47e4cf0be, api
error ExpiredTokenException: The security token included in the request is expired"
```

Kesalahan SSM dalam menjalankan perintah mesin register

Jika Anda melihat kesalahan saat mendaftarkan mesin dengan SSM, Anda mungkin perlu menjalankan ulang `nodeadm install` untuk memastikan semua dependensi SSM diinstal dengan benar.

```
"error":"running register machine command: , error: fork/exec /opt/aws/ssm-setup-cli:
no such file or directory"
```

SSM **ActivationExpired**

Saat menjalankan `nodeadm init`, jika Anda melihat kesalahan saat mendaftarkan instans dengan SSM karena aktivasi yang kedaluwarsa, Anda perlu membuat aktivasi hibrida SSM baru, memperbarui `nodeConfig.yaml` dengan `activationCode` dan `activationId` aktivasi hibrida SSM baru Anda, dan menjalankan kembali `nodeadm init`

```
"msg":"Command failed","error":"SSM activation expired. Please use a valid activation"
```

```
ERROR Registration failed due to error registering the instance with AWS SSM.
ActivationExpired
```

SSM gagal menyegarkan kredensial yang di-cache

Jika Anda melihat kegagalan untuk me-refresh kredensial cache, `/root/.aws/credentials` file tersebut mungkin telah dihapus di host Anda. Pertama periksa aktivasi hibrida SSM Anda dan pastikan aktif dan node hibrida Anda dikonfigurasi dengan benar untuk menggunakan aktivasi. Periksa log agen SSM di `/var/log/amazon/ssm` dan jalankan kembali `nodeadm init` perintah setelah Anda menyelesaikan masalah di sisi SSM.

```
"Command failed","error":"operation error SSM: DescribeInstanceInformation, get
identity: get credentials: failed to refresh cached credentials"
```

Bersihkan SSM

Untuk menghapus agen SSM dari host Anda, Anda dapat menjalankan perintah berikut.

```
dnf remove -y amazon-ssm-agent
sudo apt remove --purge amazon-ssm-agent
snap remove amazon-ssm-agent
rm -rf /var/lib/amazon/ssm/Vault/Store/RegistrationKey
```

AWS Peran IAM Di Mana Saja Pemecahan masalah

Jika Anda menggunakan AWS IAM Roles Anywhere untuk kredensial node hybrid Anda, perhatikan direktori dan artefak berikut yang diinstal pada node hybrid Anda oleh `nodeadm`. Untuk informasi selengkapnya tentang pemecahan masalah Peran IAM Di Mana Saja, lihat [Memecahkan Masalah identitas dan akses Peran AWS IAM Anywhere di mana saja di Panduan Pengguna](#) IAM Roles Anywhere. AWS

Deskripsi	Lokasi
Peran IAM Di Mana Saja CLI	<code>/usr/local/bin/aws_signing_helper</code>
Lokasi dan nama sertifikat default	<code>/etc/iam/pki/server.pem</code>
Lokasi dan nama kunci default	<code>/etc/iam/pki/server.key</code>

Peran IAM Di Mana Saja gagal menyegarkan kredensial yang di-cache

Jika Anda melihat kegagalan untuk me-refresh kredensial cache, tinjau konten `/etc/aws/hybrid/config` dan konfirmasikan bahwa IAM Roles Anywhere telah dikonfigurasi dengan benar dalam konfigurasi Anda. `nodeadm` Konfirmasikan bahwa `/etc/iam/pki` ada. Setiap node harus memiliki sertifikat dan kunci yang unik. Secara default, saat menggunakan IAM Roles Anywhere sebagai penyedia kredensial, `nodeadm` gunakan `/etc/iam/pki/server.pem` untuk lokasi dan nama sertifikat, dan `/etc/iam/pki/server.key` untuk kunci pribadi. Anda mungkin perlu membuat direktori sebelum menempatkan sertifikat dan kunci di direktori dengan `sudo mkdir -p /etc/iam/pki`. Anda dapat memverifikasi konten sertifikat Anda dengan perintah di bawah ini.

```
openssl x509 -text -noout -in server.pem
```

```
open /etc/iam/pki/server.pem: no such file or directory
could not parse PEM data
Command failed {"error": "... get identity: get credentials: failed to refresh cached
credentials, process provider error: error in credential_process: exit status 1"}
```

Peran IAM Di Mana Saja tidak diizinkan untuk melakukan **sts:AssumeRole**

Di kubelet log, jika Anda melihat masalah akses ditolak untuk `sts:AssumeRole` operasi saat menggunakan Peran IAM Di Mana Saja, periksa kebijakan kepercayaan peran IAM Hybrid Nodes Anda untuk mengonfirmasi bahwa prinsip layanan IAM Roles Anywhere diizinkan untuk

mengasumsikan Peran IAM Node Hybrid. Selain itu, konfirmasi bahwa ARN jangkar kepercayaan dikonfigurasi dengan benar dalam kebijakan kepercayaan peran IAM Hybrid Nodes Anda dan bahwa peran IAM Hybrid Nodes Anda ditambahkan ke profil IAM Roles Anywhere Anda.

```
could not get token: AccessDenied: User: ... is not authorized to perform:
sts:AssumeRole on resource: ...
```

Peran IAM Di Mana Saja tidak diizinkan untuk disetel **roleSessionName**

Di kubelet log, jika Anda melihat masalah akses ditolak untuk menyetel `roleSessionName`, konfirmasi bahwa Anda telah menyetel `acceptRoleSessionName` ke `true` untuk profil IAM Roles Anywhere Anda.

```
AccessDeniedException: Not authorized to set roleSessionName
```

Pemecahan masalah sistem operasi

RHEL

Kegagalan pendaftaran pengelola hak atau langganan

Jika Anda menjalankan `nodeadm install` dan mengalami kegagalan untuk menginstal dependensi node hibrida karena masalah pendaftaran hak, pastikan Anda telah mengatur nama pengguna dan kata sandi Red Hat dengan benar di host Anda.

```
This system is not registered with an entitlement server
```

Ubuntu

GLIBC tidak ditemukan

Jika Anda menggunakan Ubuntu untuk sistem operasi dan IAM Roles Anywhere untuk penyedia kredensi Anda dengan node hybrid dan melihat masalah dengan GLIBC tidak ditemukan, Anda dapat menginstal ketergantungan itu secara manual untuk menyelesaikan masalah.

```
GLIBC_2.32 not found (required by /usr/local/bin/aws_signing_helper)
```

Jalankan perintah berikut untuk menginstal dependensi:

```
ldd --version
```



```
sudo apt update && apt install libc6  
sudo apt install glibc-source
```

Bottlerocket

Jika Anda mengaktifkan penampung admin Bottlerocket, Anda dapat mengaksesnya dengan SSH untuk debugging lanjutan dan pemecahan masalah dengan hak istimewa yang lebih tinggi. Bagian berikut berisi perintah yang perlu dijalankan pada konteks host Bottlerocket. Setelah Anda berada di wadah admin, Anda dapat menjalankan `sheltie` untuk mendapatkan shell root penuh di host Bottlerocket.

```
sheltie
```

Anda juga dapat menjalankan perintah di bagian berikut dari shell kontainer admin dengan awalan setiap perintah dengan `sudo chroot /.bottlerocket/rootfs`.

```
sudo chroot /.bottlerocket/rootfs <command>
```

Menggunakan logdog untuk pengumpulan log

Bottlerocket menyediakan `logdog` utilitas untuk mengumpulkan log dan informasi sistem secara efisien untuk tujuan pemecahan masalah.

```
logdog
```

`logdog` utilitas mengumpulkan log dari berbagai lokasi pada host Bottlerocket dan menggabungkannya menjadi tarball. Secara default, tarball akan dibuat di `/var/log/support/bottlerocket-logs.tar.gz`, dan dapat diakses dari wadah host di `/.bottlerocket/support/bottlerocket-logs.tar.gz`.

Mengakses log sistem dengan `journalctl`

Anda dapat memeriksa status berbagai layanan sistem seperti `kubelet`, `containerd`, dll dan melihat log mereka dengan perintah berikut. `-f` Bendera akan mengikuti log secara real time.

Untuk memeriksa status `kubelet` layanan dan mengambil `kubelet` log, Anda dapat menjalankan:

```
systemctl status kubelet  
journalctl -u kubelet -f
```

Untuk memeriksa status containerd layanan dan mengambil log untuk containerd instance yang diatur, Anda dapat menjalankan:

```
systemctl status containerd  
journalctl -u containerd -f
```

Untuk memeriksa status host-containerd layanan dan mengambil log untuk containerd instance host, Anda dapat menjalankan:

```
systemctl status host-containerd  
journalctl -u host-containerd -f
```

Untuk mengambil log untuk wadah bootstrap dan wadah host, Anda dapat menjalankan:

```
journalctl _COMM=host-ctr -f
```

Gunakan penyimpanan data aplikasi untuk kluster Anda

Anda dapat menggunakan berbagai layanan AWS penyimpanan dengan Amazon EKS untuk kebutuhan penyimpanan aplikasi Anda. Melalui driver Container Storage Interface (CSI) yang AWS didukung, Anda dapat dengan mudah menggunakan Amazon EBS, Amazon S3, Amazon EFS, Amazon FSx, dan Amazon File Cache untuk kebutuhan penyimpanan aplikasi Anda yang berjalan di Amazon EKS.

Bab ini mencakup pilihan penyimpanan untuk kluster Amazon EKS.

Topik

- [Gunakan penyimpanan volume Kubernetes dengan Amazon EBS](#)
- [Gunakan penyimpanan sistem file elastis dengan Amazon EFS](#)
- [Gunakan penyimpanan aplikasi berkinerja tinggi dengan Amazon FSx untuk Lustre](#)
- [Gunakan penyimpanan aplikasi berkinerja tinggi FSx untuk NetApp ONTAP](#)
- [Gunakan penyimpanan data dengan Amazon FSx untuk OpenZFS](#)
- [Minimalkan latensi dengan Amazon File Cache](#)
- [Akses objek Amazon S3 dengan Mountpoint untuk driver Amazon S3 CSI](#)
- [Aktifkan fungsionalitas snapshot untuk volume CSI](#)

Gunakan penyimpanan volume Kubernetes dengan Amazon EBS

Note

Baru: Amazon EKS Auto Mode mengotomatiskan tugas rutin untuk penyimpanan blok. Pelajari cara [the section called “Menyebarkan beban kerja stateful”](#).

Driver [Amazon Elastic Block Store \(Amazon EBS\) Container Storage Interface \(CSI\)](#) mengelola siklus hidup volume Amazon EBS sebagai penyimpanan untuk Volume Kubernetes yang Anda buat. [Driver Amazon EBS CSI membuat volume Amazon EBS untuk jenis volume Kubernetes ini: volume fana generik dan volume persisten.](#)

Pertimbangan

- Anda tidak perlu menginstal pengontrol Amazon EBS CSI pada kluster Mode Otomatis EKS.
- Anda tidak dapat memasang volume Amazon EBS ke Pod Fargate.
- Anda dapat menjalankan pengontrol Amazon EBS CSI di node Fargate, tetapi node Amazon EBS CSI DaemonSet hanya dapat berjalan di instans Amazon. EC2
- Volume Amazon EBS dan driver Amazon EBS CSI tidak kompatibel dengan Amazon EKS Hybrid Nodes.
- Support akan disediakan untuk versi add-on terbaru dan satu versi sebelumnya. Bug atau kerentanan yang ditemukan di versi terbaru akan di-backport ke rilis sebelumnya dalam versi minor baru.
- Mode Otomatis EKS membutuhkan kelas penyimpanan untuk digunakan `ebs.csi.eks.amazonaws.com` sebagai penyedia. Amazon EBS CSI Driver (`ebs.csi.aws.com`) standar mengelola volumenya sendiri secara terpisah. Untuk menggunakan volume yang ada dengan Mode Otomatis EKS, migrasikan volume menggunakan snapshot volume ke kelas penyimpanan yang menggunakan penyedia Mode Otomatis.

Important

Untuk menggunakan fungsionalitas snapshot dari driver Amazon EBS CSI, Anda harus menginstal pengontrol snapshot CSI terlebih dahulu. Untuk informasi selengkapnya, lihat [the section called “Pengontrol snapshot CSI”](#).

Prasyarat

- Sebuah kluster yang sudah ada. Untuk melihat versi platform yang diperlukan, jalankan perintah berikut.

```
aws eks describe-addon-versions --addon-name aws-ebs-csi-driver
```

- Driver EBS CSI membutuhkan AWS Izin IAM.
 - AWS menyarankan menggunakan EKS Pod Identities. Untuk informasi selengkapnya, lihat [the section called “Ikhtisar pengaturan EKS Pod Identities”](#).

- Untuk informasi tentang Peran IAM untuk Akun Layanan, lihat [the section called “Penyedia IAM OIDC”](#).

Langkah 1: Buat peran IAM

Plugin Amazon EBS CSI memerlukan izin IAM untuk melakukan panggilan atas nama Anda. AWS APIs Jika Anda tidak melakukan langkah-langkah ini, mencoba menginstal add-on dan menjalankan `kubectl describe pvc` akan muncul `failed to provision volume with StorageClass` bersama dengan `could not create volume in EC2: UnauthorizedOperation` kesalahan. Untuk informasi selengkapnya, lihat [Mengatur izin pengemudi](#) di GitHub.

Note

Pod akan memiliki akses ke izin yang ditetapkan untuk peran IAM kecuali Anda memblokir akses ke IMDS. Untuk informasi selengkapnya, lihat [the section called “Praktik terbaik”](#).

Prosedur berikut menunjukkan cara membuat peran IAM dan melampirkan kebijakan AWS terkelola padanya. Untuk menerapkan prosedur ini, Anda dapat menggunakan salah satu alat ini:

- [the section called “eksctl”](#)
- [the section called “AWS Management Console”](#)
- [the section called “AWS CLI”](#)

Note

Langkah-langkah spesifik dalam prosedur ini ditulis untuk menggunakan driver sebagai add-on Amazon EKS. Langkah-langkah yang berbeda diperlukan untuk menggunakan driver sebagai add-on yang dikelola sendiri. Untuk informasi selengkapnya, lihat [Mengatur izin driver](#) pada GitHub.

eksctl

1. Buat peran IAM dan lampirkan kebijakan. AWS mempertahankan kebijakan AWS terkelola atau Anda dapat membuat kebijakan kustom Anda sendiri. Anda dapat membuat peran IAM dan

melampirkan kebijakan AWS terkelola dengan perintah berikut. Ganti *my-cluster* dengan nama kluster Anda. Perintah menyebarkan AWS CloudFormation tumpukan yang membuat peran IAM dan melampirkan kebijakan IAM padanya.

```
eksctl create iamserviceaccount \  
    --name ebs-csi-controller-sa \  
    --namespace kube-system \  
    --cluster my-cluster \  
    --role-name AmazonEKS_EBS_CSI_DriverRole \  
    --role-only \  
    --attach-policy-arn arn:aws:iam::aws:policy/service-role/  
AmazonEBSCSIDriverPolicy \  
    --approve
```

2. Anda dapat melewati langkah ini jika Anda tidak menggunakan [kunci KMS](#) khusus. Jika Anda menggunakannya untuk enkripsi pada volume Amazon EBS Anda, sesuaikan peran IAM sesuai kebutuhan. Sebagai contoh, lakukan hal berikut:
 - a. Salin dan tempel kode berikut ke `kms-key-for-encryption-on-ebs.json` file baru. Ganti *custom-key-arn* dengan [ARN kunci KMS](#) kustom.

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "kms:CreateGrant",  
        "kms:ListGrants",  
        "kms:RevokeGrant"  
      ],  
      "Resource": ["custom-key-arn"],  
      "Condition": {  
        "Bool": {  
          "kms:GrantIsForAWSResource": "true"  
        }  
      }  
    },  
    {  
      "Effect": "Allow",  
      "Action": [  
        "kms:Encrypt",  
        "kms:Decrypt",  
        "kms:DescribeKey"  
      ],  
      "Resource": ["custom-key-arn"]  
    }  
  ]  
}
```

```
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
    ],
    "Resource": ["custom-key-arn"]
}
]
```

- b. Buat kebijakan. Anda dapat mengubah *KMS_Key_For_Encryption_On_EBS_Policy* ke nama yang berbeda. Namun, jika Anda melakukannya, pastikan untuk mengubahnya di langkah selanjutnya juga.

```
aws iam create-policy \
    --policy-name KMS_Key_For_Encryption_On_EBS_Policy \
    --policy-document file://kms-key-for-encryption-on-ebs.json
```

- c. Lampirkan kebijakan IAM ke peran dengan perintah berikut. Ganti *111122223333* dengan ID akun Anda.

```
aws iam attach-role-policy \
    --policy-arn arn:aws:iam::111122223333:policy/
KMS_Key_For_Encryption_On_EBS_Policy \
    --role-name AmazonEKS_EBS_CSI_DriverRole
```

AWS Management Console

1. Buka konsol IAM di <https://console.aws.amazon.com/iam/>.
2. Di panel navigasi sebelah kiri, pilih Peran.
3. Pada halaman Peran, pilih Buat peran.
4. Pada halaman Pilih entitas tepercaya, lakukan hal berikut:
 - a. Di bagian Jenis entitas tepercaya, pilih Identitas web.
 - b. Untuk penyedia Identity, pilih URL penyedia OpenID Connect untuk klaster Anda (seperti yang ditunjukkan di bawah Ikhtisar di Amazon EKS).
 - c. Untuk Audiens, pilih `sts.amazonaws.com`.
 - d. Pilih Berikutnya.
5. Pada halaman Tambahkan izin, lakukan hal berikut:

- a. Di dalam kotak Filter kebijakan, masukkan AmazonEBSCSIDriverPolicy.
 - b. Pilih kotak centang di sebelah kiri yang AmazonEBSCSIDriverPolicy dikembalikan dalam pencarian.
 - c. Pilih Berikutnya.
6. Pada halaman Nama, tinjau, dan buat, lakukan hal berikut:
- a. Untuk nama Peran, masukkan nama unik untuk peran Anda, seperti *AmazonEKS_EBS_CSI_DriverRole*.
 - b. Di bawah Tambahkan tag (Opsional), tambahkan metadata ke peran dengan melampirkan tag sebagai pasangan nilai kunci. Untuk informasi selengkapnya tentang penggunaan tanda di IAM, lihat [Menandai sumber daya IAM](#) di Panduan Pengguna IAM.
 - c. Pilih Buat peran.
7. Setelah peran dibuat, pilih peran di konsol untuk dibuka, dan kemudian diedit.
8. Pilih tab Trust relationship, lalu pilih Edit trust policy.
9. Temukan garis yang terlihat mirip dengan baris berikut:

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud":  
"sts.amazonaws.com"
```

Tambahkan koma ke akhir baris sebelumnya, lalu tambahkan baris berikut setelah baris sebelumnya. Ganti *region-code* dengan AWS Wilayah tempat kluster Anda berada. Ganti *EXAMPLED539D4633E53DE1B71EXAMPLE* dengan ID penyedia OIDC cluster Anda.

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub":  
"system:serviceaccount:kube-system:ebs-csi-controller-sa"
```

10. Pilih Perbarui kebijakan untuk menyelesaikan.
11. Jika Anda menggunakan [kunci KMS](#) khusus untuk enkripsi pada volume Amazon EBS Anda, sesuaikan peran IAM sesuai kebutuhan. Sebagai contoh, lakukan hal berikut:
- a. Di panel navigasi di sebelah kiri, pilih Kebijakan.
 - b. Pada halaman Kebijakan, pilih Buat Kebijakan.
 - c. Di halaman Buat kebijakan, pilih tab JSON.
 - d. Salin dan tempel kode berikut ke editor, ganti *custom-key-arn* dengan [ARN kunci KMS](#) khusus.


```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:CreateGrant",
        "kms:ListGrants",
        "kms:RevokeGrant"
      ],
      "Resource": ["custom-key-arn"],
      "Condition": {
        "Bool": {
          "kms:GrantIsForAWSResource": "true"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": ["custom-key-arn"]
    }
  ]
}
```

- e. Pilih Berikutnya: Tanda.
- f. Pada halaman Tambahkan tag (Opsional), pilih Berikutnya: Ulasan.
- g. Untuk Nama, masukkan nama unik untuk kebijakan Anda (misalnya, *KMS_Key_For_Encryption_On_EBS_Policy*).
- h. Pilih Buat kebijakan.
- i. Di panel navigasi sebelah kiri, pilih Peran.
- j. Pilih **AmazonEKS_EBS_CSI_DriverRole** di konsol untuk membukanya untuk diedit.
- k. Dari daftar tarik-turun Tambahkan izin, pilih Lampirkan kebijakan.

- l. Di dalam kotak Filter kebijakan, masukkan *KMS_Key_For_Encryption_On_EBS_Policy*.
- m. Pilih kotak centang di sebelah kiri *KMS_Key_For_Encryption_On_EBS_Policy* yang dikembalikan dalam pencarian.
- n. Pilih Lampirkan kebijakan.

AWS CLI

1. Lihat URL penyedia OIDC kluster Anda. Ganti *my-cluster* dengan nama kluster Anda. Jika output dari perintah adalah None, tinjau Prasyarat.

```
aws eks describe-cluster --name my-cluster --query "cluster.identity.oidc.issuer" --output text
```

Contoh output adalah sebagai berikut.

```
https://oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE
```

2. Buat peran IAM, berikan tindakan. AssumeRoleWithWebIdentity
 - a. Salin konten berikut ke file yang diberi nama *aws-ebs-csi-driver-trust-policy.json*. Ganti *111122223333* dengan ID akun Anda. Ganti *EXAMPLED539D4633E53DE1B71EXAMPLE* dan *region-code* dengan nilai yang dikembalikan pada langkah sebelumnya.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:oidc-provider/oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com",
          "oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-system:ebs-csi-controller-sa"
        }
      }
    }
  ]
}
```

```

    }
  }
}
]
}

```

- b. Buat peran. Anda dapat mengubah *AmazonEKS_EBS_CSI_DriverRole* ke nama yang berbeda. Jika Anda mengubahnya, pastikan untuk mengubahnya di langkah selanjutnya.

```

aws iam create-role \
  --role-name AmazonEKS_EBS_CSI_DriverRole \
  --assume-role-policy-document file://"aws-ebs-csi-driver-trust-policy.json"

```

3. Lampirkan kebijakan. AWS mempertahankan kebijakan AWS terkelola atau Anda dapat membuat kebijakan kustom Anda sendiri. Lampirkan kebijakan AWS terkelola ke peran dengan perintah berikut.

```

aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy \
  --role-name AmazonEKS_EBS_CSI_DriverRole

```

4. Jika Anda menggunakan [kunci KMS](#) khusus untuk enkripsi pada volume Amazon EBS Anda, sesuaikan peran IAM sesuai kebutuhan. Sebagai contoh, lakukan hal berikut:

- a. Salin dan tempel kode berikut ke `kms-key-for-encryption-on-ebs.json` file baru. Ganti *custom-key-arn* dengan [ARN kunci KMS](#) kustom.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:CreateGrant",
        "kms:ListGrants",
        "kms:RevokeGrant"
      ],
      "Resource": ["custom-key-arn"],
      "Condition": {
        "Bool": {
          "kms:GrantIsForAWSResource": "true"
        }
      }
    }
  ]
}

```

```

    },
    {
      "Effect": "Allow",
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": ["custom-key-arn"]
    }
  ]
}

```

- b. Buat kebijakan. Anda dapat mengubah *KMS_Key_For_Encryption_On_EBS_Policy* ke nama yang berbeda. Namun, jika Anda melakukannya, pastikan untuk mengubahnya di langkah selanjutnya juga.

```

aws iam create-policy \
  --policy-name KMS_Key_For_Encryption_On_EBS_Policy \
  --policy-document file://kms-key-for-encryption-on-ebs.json

```

- c. Lampirkan kebijakan IAM ke peran dengan perintah berikut. Ganti *111122223333* dengan ID akun Anda.

```

aws iam attach-role-policy \
  --policy-arn arn:aws:iam::111122223333:policy/
  KMS_Key_For_Encryption_On_EBS_Policy \
  --role-name AmazonEKS_EBS_CSI_DriverRole

```

Sekarang setelah Anda membuat peran IAM driver Amazon EBS CSI, Anda dapat melanjutkan ke bagian berikutnya. Saat Anda menerapkan add-on dengan peran IAM ini, ia membuat dan dikonfigurasi untuk menggunakan akun layanan yang diberi nama *ebs-csi-controller-sa* Akun layanan terikat pada Kubernetes yang diberi izin Kubernetes *clusterrole* yang diperlukan.

Langkah 2: Dapatkan driver Amazon EBS CSI

Kami menyarankan Anda menginstal driver Amazon EBS CSI melalui add-on Amazon EKS untuk meningkatkan keamanan dan mengurangi jumlah pekerjaan. Untuk menambahkan add-on Amazon

EKS ke cluster Anda, lihat [the section called “Buat add-on”](#). Untuk informasi selengkapnya tentang add-on, lihat [the section called “Add-on Amazon EKS”](#).

⚠ Important

Sebelum menambahkan driver Amazon EBS sebagai add-on Amazon EKS, konfirmasi bahwa Anda tidak memiliki versi driver yang dikelola sendiri yang diinstal pada cluster Anda. Jika demikian, lihat [Menghapus instalasi driver Amazon EBS CSI yang dikelola sendiri](#).
GitHub

[Atau, jika Anda ingin instalasi yang dikelola sendiri dari driver Amazon EBS CSI, lihat Instalasi aktif](#).
GitHub

Langkah 3: Menyebarkan aplikasi sampel

Anda dapat menerapkan berbagai contoh aplikasi dan memodifikasinya sesuai kebutuhan. Untuk informasi selengkapnya, lihat Contoh [Kubernetes di](#). GitHub

Gunakan penyimpanan sistem file elastis dengan Amazon EFS

[Amazon Elastic File System](#) (Amazon EFS) menyediakan penyimpanan file tanpa server dan sepenuhnya elastis sehingga Anda dapat berbagi data file tanpa menyediakan atau mengelola kapasitas dan kinerja penyimpanan. [Driver Amazon EFS Container Storage Interface \(CSI\)](#) menyediakan antarmuka CSI yang memungkinkan cluster Kubernetes berjalan untuk AWS mengelola siklus hidup sistem file Amazon EFS. Topik ini menunjukkan cara menerapkan driver Amazon EFS CSI ke cluster Amazon EKS Anda.

Pertimbangan

- Driver Amazon EFS CSI tidak kompatibel dengan gambar kontainer berbasis Windows.
- [Anda tidak dapat menggunakan penyediaan dinamis untuk volume persisten dengan node Fargate, tetapi Anda dapat menggunakan penyediaan statis](#).
- [Penyediaan dinamis](#) membutuhkan [1.2](#) atau lebih baru dari driver. Anda dapat menggunakan [penyediaan statis](#) untuk volume persisten menggunakan versi 1.1 driver pada versi kluster Amazon EKS yang didukung (lihat [the section called “Versi Kubernetes”](#)).
- Versi [1.3.2](#) atau yang lebih baru dari driver ini mendukung arsitektur Arm64, termasuk instance berbasis Amazon EC2 Graviton.

- Versi [1.4.2](#) atau yang lebih baru dari driver ini mendukung penggunaan FIPS untuk memasang sistem file.
- Catat kuota sumber daya untuk Amazon EFS. Misalnya, ada kuota 1000 titik akses yang dapat dibuat untuk setiap sistem file Amazon EFS. Untuk informasi selengkapnya, lihat [Kuota sumber daya Amazon EFS yang tidak dapat Anda ubah](#).
- Mulai versi [2.0.0](#), driver ini beralih dari menggunakan `stunnel` ke `efs-proxy` koneksi TLS. Ketika `efs-proxy` digunakan, itu akan membuka sejumlah thread yang sama dengan satu ditambah jumlah core untuk node yang sedang berjalan.
- Driver Amazon EFS CSI tidak kompatibel dengan Amazon EKS Hybrid Nodes.

Prasyarat

- Driver Amazon EFS CSI memerlukan izin AWS Identity and Access Management (IAM).
 - AWS menyarankan menggunakan EKS Pod Identities. Untuk informasi selengkapnya, lihat [the section called “Ikhtisar pengaturan EKS Pod Identities”](#).
 - Untuk informasi tentang peran IAM untuk akun layanan dan menyiapkan penyedia IAM OpenID Connect (OIDC) untuk kluster Anda, lihat [the section called “Penyedia IAM OIDC”](#)
- Versi 2.12.3 atau yang lebih baru atau versi 1.27.160 atau yang lebih baru dari AWS Command Line Interface (AWS CLI) diinstal dan dikonfigurasi pada perangkat Anda atau AWS CloudShell. Untuk memeriksa versi Anda saat ini, gunakan `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Package manager seperti `yum` atau `apt-get`, atau Homebrew untuk macOS seringkali merupakan beberapa versi di belakang versi terbaru CLI. AWS Untuk menginstal versi terbaru, lihat [Menginstal](#) dan [Konfigurasi cepat dengan aws configure](#) di Panduan Pengguna Antarmuka Baris Perintah. Versi AWS CLI yang diinstal AWS CloudShell mungkin juga beberapa versi di belakang versi terbaru. Untuk memperbaruinya, lihat [Menginstal AWS CLI ke direktori home Anda](#) di AWS CloudShell Panduan Pengguna.
- Alat baris perintah `kubectl` diinstal pada perangkat Anda atau AWS CloudShell. Versinya bisa sama dengan atau hingga satu versi minor lebih awal atau lebih lambat dari versi Kubernetes dari kluster Anda. Misalnya, jika versi cluster Anda 1.29, Anda dapat menggunakan `kubectl` versi 1.28, 1.29, atau 1.30 dengan itu. Untuk menginstal atau memutakhirkan `kubectl`, lihat [the section called “Siapkan kubectl dan eksctl”](#).

Note

Pod yang berjalan di Fargate secara otomatis memasang sistem file Amazon EFS, tanpa memerlukan langkah penginstalan driver secara manual.

Langkah 1: Buat peran IAM

Driver Amazon EFS CSI memerlukan izin IAM untuk berinteraksi dengan sistem file Anda. Buat peran IAM dan lampirkan kebijakan AWS terkelola yang diperlukan padanya. Untuk menerapkan prosedur ini, Anda dapat menggunakan salah satu alat ini:

- [the section called “eksctl”](#)
- [the section called “AWS Management Console”](#)
- [the section called “AWS CLI”](#)

Note

Langkah-langkah spesifik dalam prosedur ini ditulis untuk menggunakan driver sebagai add-on Amazon EKS. Untuk detail tentang penginstalan yang dikelola sendiri, lihat [Mengatur izin driver](#) pada. GitHub

eksctl

Jika menggunakan Identitas Pod

Jalankan perintah berikut untuk membuat peran IAM dan asosiasi Pod Identity dengan `eksctl`. Ganti `my-cluster` dengan nama kluster Anda. Anda juga dapat mengganti `AmazonEKS_EFS_CSI_DriverRole` dengan nama yang berbeda.

```
export cluster_name=my-cluster
export role_name=AmazonEKS_EFS_CSI_DriverRole
eksctl create podidentityassociation \
  --service-account-name efs-csi-controller-sa \
  --namespace kube-system \
  --cluster $cluster_name \
  --role-name $role_name \
```

```
--permission-policy-arns arn:aws: iam::aws:policy/service-role/  
AmazonEFSCSIDriverPolicy
```

Jika menggunakan peran IAM untuk akun layanan

Jalankan perintah berikut untuk membuat peran IAM dengan `eksctl`. Ganti `my-cluster` dengan nama kluster Anda. Anda juga dapat mengganti `AmazonEKS_EFS_CSI_DriverRole` dengan nama yang berbeda.

```
export cluster_name=my-cluster  
export role_name=AmazonEKS_EFS_CSI_DriverRole  
eksctl create iamserviceaccount \  
    --name efs-csi-controller-sa \  
    --namespace kube-system \  
    --cluster $cluster_name \  
    --role-name $role_name \  
    --role-only \  
    --attach-policy-arn arn:aws: iam::aws:policy/service-role/AmazonEFSCSIDriverPolicy \  
    --approve  
TRUST_POLICY=$(aws iam get-role --output json --role-name $role_name --query  
    'Role.AssumeRolePolicyDocument' | \  
    sed -e 's/efs-csi-controller-sa/efs-csi-*/' -e 's/StringEquals/StringLike/')  
aws iam update-assume-role-policy --role-name $role_name --policy-document  
"$TRUST_POLICY"
```

AWS Management Console

Jalankan berikut ini untuk membuat peran IAM dengan AWS Management Console.

1. Buka konsol IAM di <https://console.aws.amazon.com/iam/>.
2. Di panel navigasi sebelah kiri, pilih Peran.
3. Pada halaman Peran, pilih Buat peran.
4. Pada halaman Pilih entitas tepercaya, lakukan hal berikut:
 - a. Jika menggunakan Identitas Pod EKS:
 - i. Di bagian Jenis entitas tepercaya, pilih AWS layanan.
 - ii. Dalam tarik-turun Layanan atau kasus penggunaan, pilih EKS.
 - iii. Di bagian Use case, pilih EKS - Pod Identity.
 - iv. Pilih Berikutnya.

- b. Jika menggunakan peran IAM untuk akun layanan:
 - i. Di bagian Jenis entitas tepercaya, pilih Identitas web.
 - ii. Untuk penyedia Identity, pilih URL penyedia OpenID Connect untuk klaster Anda (seperti yang ditunjukkan di bawah Ikhtisar di Amazon EKS).
 - iii. Untuk Audiens, pilih `sts.amazonaws.com`.
 - iv. Pilih Berikutnya.
5. Pada halaman Tambahkan izin, lakukan hal berikut:
 - a. Di dalam kotak Filter kebijakan, masukkan `AmazonEFSCSIDriverPolicy`.
 - b. Pilih kotak centang di sebelah kiri yang `AmazonEFSCSIDriverPolicy` dikembalikan dalam pencarian.
 - c. Pilih Berikutnya.
6. Pada halaman Nama, tinjau, dan buat, lakukan hal berikut:
 - a. Untuk nama Peran, masukkan nama unik untuk peran Anda, seperti `AmazonEKS_EFS_CSI_DriverRole`.
 - b. Di bawah Tambahkan tag (Opsional), tambahkan metadata ke peran dengan melampirkan tag sebagai pasangan nilai kunci. Untuk informasi selengkapnya tentang penggunaan tanda di IAM, lihat [Menandai sumber daya IAM](#) di Panduan Pengguna IAM.
 - c. Pilih Buat peran.
7. Setelah peran dibuat:
 - a. Jika menggunakan Identitas Pod EKS:
 - i. Buka [konsol Amazon EKS](#).
 - ii. Di panel navigasi kiri, pilih Clusters, lalu pilih nama cluster yang ingin Anda konfigurasi untuk asosiasi EKS Pod Identity.
 - iii. Pilih tab Access.
 - iv. Dalam asosiasi Pod Identity, pilih Create.
 - v. Pilih dropdown peran IAM dan pilih peran yang baru Anda buat.
 - vi. Pilih kolom namespace Kubernetes dan masukan. `kube-system`
 - vii. Pilih kolom akun layanan Kubernetes dan masukan. `efs-csi-controller-sa`
 - viii. Pilih Buat.
 - ix. Untuk informasi selengkapnya tentang membuat asosiasi Pod Identity, lihat [the section called "Membuat asosiasi Pod Identity \(AWS Console\)"](#).

- b. Jika menggunakan peran IAM untuk akun layanan:
 - i. Pilih peran untuk membukanya untuk diedit.
 - ii. Pilih tab Trust relationship, lalu pilih Edit trust policy.
 - iii. Temukan garis yang terlihat mirip dengan baris berikut:

```
"oidc.eks.region-code.amazonaws.com/id/<EXAMPLED539D4633E53DE1B71EXAMPLE>:aud":
  "sts.amazonaws.com"
```

Tambahkan baris berikut di atas baris sebelumnya. Ganti <region-code> dengan AWS Wilayah tempat cluster Anda berada. Ganti <EXAMPLED539D4633E53DE1B71EXAMPLE> dengan ID penyedia OIDC cluster Anda.

```
"oidc.eks.<region-code>.amazonaws.com/id/
<EXAMPLED539D4633E53DE1B71EXAMPLE>:sub": "system:serviceaccount:kube-system:efs-
csi-*",
```

- iv. Ubah Condition operator dari "StringEquals" ke "StringLike".
- v. Pilih Perbarui kebijakan untuk menyelesaikan.

AWS CLI

Jalankan perintah berikut untuk membuat peran IAM dengan AWS CLI.

Jika menggunakan Identitas Pod

1. Buat peran IAM yang memberikan AssumeRole dan TagSession tindakan ke layanan.


```
pods.eks.amazonaws.com
```

 - a. Salin isi berikut ke file bernama `aws-efs-csi-driver-trust-policy-pod-identity.json`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
      "Effect": "Allow",
      "Principal": {
        "Service": "pods.eks.amazonaws.com"
```

```

    },
    "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
    ]
}
]
}

```

- b. Buat peran. Ganti `my-cluster` dengan nama kluster Anda. Anda juga dapat mengganti `AmazonEKS_EFS_CSI_DriverRole` dengan nama yang berbeda.

```

export cluster_name=my-cluster
export role_name=AmazonEKS_EFS_CSI_DriverRole
aws iam create-role \
  --role-name $role_name \
  --assume-role-policy-document file://"aws-efs-csi-driver-trust-policy-pod-identity.json"

```

2. Lampirkan kebijakan AWS terkelola yang diperlukan ke peran dengan perintah berikut.

```

aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/service-role/AmazonEFSCSIDriverPolicy \
  --role-name $role_name

```

3. Jalankan perintah berikut untuk membuat asosiasi Pod Identity. Ganti `arn:aws:iam::<111122223333>:role/my-role` dengan peran yang dibuat pada langkah sebelumnya.

```

aws eks create-pod-identity-association --cluster-name $cluster_name --role-arn {arn-aws}iam::<111122223333>:role/my-role --namespace kube-system --service-account efs-csi-controller-sa

```

4. Untuk informasi selengkapnya tentang membuat asosiasi Pod Identity, lihat [the section called "Membuat asosiasi Pod Identity \(AWS Console\)"](#).

Jika menggunakan peran IAM untuk akun layanan

1. Lihat URL penyedia OIDC kluster Anda. Ganti `my-cluster` dengan nama kluster Anda. Anda juga dapat mengganti `AmazonEKS_EFS_CSI_DriverRole` dengan nama yang berbeda.

```

export cluster_name=my-cluster

```

```
export role_name=AmazonEKS_EFS_CSI_DriverRole
aws eks describe-cluster --name $cluster_name --query "cluster.identity.oidc.issuer"
--output text
```

Contoh output adalah sebagai berikut.

```
https://oidc.eks.<region-code>.amazonaws.com/id/<EXAMPLED539D4633E53DE1B71EXAMPLE>
```

Jika output dari perintah adalah None, tinjau Prasyarat.

2. Buat peran IAM yang memberikan tindakan. AssumeRoleWithWebIdentity

- a. Salin isi berikut ke file bernama `aws-efs-csi-driver-trust-policy.json`. Ganti `<111122223333>` dengan ID akun Anda. Ganti `<EXAMPLED539D4633E53DE1B71EXAMPLE>` dan `<region-code>` dengan nilai yang dikembalikan pada langkah sebelumnya.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::<111122223333>:oidc-provider/oidc.eks.<region-code>.amazonaws.com/id/<EXAMPLED539D4633E53DE1B71EXAMPLE>"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringLike": {
          "oidc.eks.region-code.amazonaws.com/id/<EXAMPLED539D4633E53DE1B71EXAMPLE>:sub": "system:serviceaccount:kube-system:efs-csi-*",
          "oidc.eks.region-code.amazonaws.com/id/<EXAMPLED539D4633E53DE1B71EXAMPLE>:aud": "sts.amazonaws.com"
        }
      }
    }
  ]
}
```

- b. Buat peran.

```
aws iam create-role \
  --role-name $role_name \
```

```
--assume-role-policy-document file://"aws-efs-csi-driver-trust-policy.json"
```

3. Lampirkan kebijakan AWS terkelola yang diperlukan ke peran dengan perintah berikut.

```
aws iam attach-role-policy \  
  --policy-arn arn:aws:iam::aws:policy/service-role/AmazonEFSCSIDriverPolicy \  
  --role-name $role_name
```

Langkah 2: Dapatkan driver Amazon EFS CSI

Kami menyarankan Anda menginstal driver Amazon EFS CSI melalui add-on Amazon EKS. Untuk menambahkan add-on Amazon EKS ke cluster Anda, lihat [the section called “Buat add-on”](#). Untuk informasi selengkapnya tentang add-on, lihat [the section called “Add-on Amazon EKS”](#). Jika Anda tidak dapat menggunakan add-on Amazon EKS, kami mendorong Anda untuk mengirimkan masalah tentang mengapa Anda tidak dapat ke repositori [peta jalan GitHub Containers](#).

Important

Sebelum menambahkan driver Amazon EFS sebagai add-on Amazon EKS, konfirmasi bahwa Anda tidak memiliki versi driver yang dikelola sendiri yang diinstal di cluster Anda. Jika demikian, lihat [Menghapus Instalasi Driver Amazon EFS CSI](#) di GitHub.

Atau, jika Anda ingin instalasi yang dikelola sendiri dari driver Amazon EFS CSI, lihat [Instalasi](#) aktif GitHub.

Langkah 3: Buat sistem file Amazon EFS

Untuk membuat sistem file Amazon EFS, lihat [Membuat sistem file Amazon EFS untuk Amazon EKS](#) di GitHub.

Langkah 4: Menyebarkan aplikasi sampel

Anda dapat menerapkan berbagai contoh aplikasi dan memodifikasinya sesuai kebutuhan. Untuk informasi selengkapnya, lihat [Contoh](#) di GitHub.

Gunakan penyimpanan aplikasi berkinerja tinggi dengan Amazon FSx untuk Lustre

[Driver Amazon FSx for Lustre Container Storage Interface \(CSI\)](#) menyediakan antarmuka CSI yang memungkinkan kluster Amazon EKS mengelola siklus hidup Amazon untuk sistem file Lustre. FSx Untuk informasi selengkapnya, lihat [Panduan Pengguna Amazon FSx for Lustre](#).

Untuk detail tentang cara menerapkan driver Amazon FSx for Lustre CSI ke kluster Amazon EKS Anda dan memverifikasi apakah itu berfungsi, lihat [the section called “Menyebarkan driver”](#)

Menyebarkan driver FSx untuk Lustre

Topik ini menunjukkan cara menerapkan driver [FSx for Lustre CSI ke](#) cluster Amazon EKS Anda dan memverifikasi bahwa itu berfungsi. Kami merekomendasikan menggunakan versi driver terbaru. Untuk versi yang tersedia, lihat [Matriks Kompatibilitas Spesifikasi CSI](#) pada GitHub.

Note

Driver tidak didukung di Fargate atau Amazon EKS Hybrid Nodes.

Untuk deskripsi rinci tentang parameter yang tersedia dan contoh lengkap yang menunjukkan fitur pengemudi, lihat proyek driver [FSx untuk Lustre Container Storage Interface \(CSI\)](#) di GitHub

Prasyarat

- Sebuah klaster yang sudah ada.
- Add-on Amazon FSx CSI Driver EKS memerlukan agen EKS Pod Identity untuk otentikasi. Tanpa komponen ini, add-on akan gagal dengan kesalahan `Amazon EKS Pod Identity agent is not installed in the cluster`, mencegah operasi volume. Instal agen Pod Identity sebelum atau sesudah menerapkan add-on Driver FSx CSI. Untuk informasi selengkapnya, lihat [the section called “Mengatur Agen”](#).
- Versi 2.12.3 atau yang lebih baru atau versi 1.27.160 atau yang lebih baru dari AWS Command Line Interface (AWS CLI) diinstal dan dikonfigurasi pada perangkat Anda atau AWS CloudShell. Untuk memeriksa versi Anda saat ini, gunakan `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Package manager seperti `yum` atau `apt-get`, atau Homebrew untuk macOS seringkali merupakan beberapa versi di belakang versi terbaru CLI. AWS Untuk menginstal versi terbaru, lihat [Menginstal](#) dan [Konfigurasi cepat dengan aws configure](#) di Panduan Pengguna Antarmuka

Baris AWS Perintah. Versi AWS CLI yang diinstal AWS CloudShell mungkin juga beberapa versi di belakang versi terbaru. Untuk memperbaruinya, lihat [Menginstal AWS CLI ke direktori home Anda](#) di AWS CloudShell Panduan Pengguna.

- Versi 0.210.0 atau yang lebih baru dari alat baris `eksctl` perintah yang diinstal pada perangkat Anda atau AWS CloudShell. Untuk menginstal atau memperbaruieksctl, lihat [Instalasi](#) dalam eksctl dokumentasi.
- Alat baris `kubectl` perintah diinstal pada perangkat Anda atau AWS CloudShell. Versinya bisa sama dengan atau hingga satu versi minor lebih awal atau lebih lambat dari versi Kubernetes dari kluster Anda. Misalnya, jika versi cluster Anda 1.29, Anda dapat menggunakan `kubectl` versi 1.28, 1.29, atau 1.30 dengan itu. Untuk menginstal atau memutakhirkan `kubectl`, lihat [the section called “Siapkan kubectl dan eksctl”](#).

Langkah 1: Buat peran IAM

Plugin Amazon FSx CSI memerlukan izin IAM untuk melakukan panggilan AWS APIs atas nama Anda.

Note

Pod akan memiliki akses ke izin yang ditetapkan ke peran IAM kecuali Anda memblokir akses ke IMDS. Untuk informasi selengkapnya, lihat [the section called “Praktik terbaik”](#).

Prosedur berikut menunjukkan cara membuat peran IAM dan melampirkan kebijakan AWS terkelola padanya.

1. Buat peran IAM dan lampirkan kebijakan AWS terkelola dengan perintah berikut. Ganti `my-cluster` dengan nama cluster yang ingin Anda gunakan. Perintah menyebarkan AWS CloudFormation tumpukan yang membuat peran IAM dan melampirkan kebijakan IAM padanya.

```
eksctl create iamserviceaccount \
  --name fsx-csi-controller-sa \
  --namespace kube-system \
  --cluster my-cluster \
  --role-name AmazonEKS_FSx_CSI_DriverRole \
  --role-only \
  --attach-policy-arn arn:aws:iam::aws:policy/AmazonFSxFullAccess \
  --approve
```

Anda akan melihat beberapa baris output saat akun layanan dibuat. Baris output terakhir mirip dengan yang berikut ini.

```
[#] 1 task: {
    2 sequential sub-tasks: {
        create IAM role for serviceaccount "kube-system/fsx-csi-controller-sa",
        create serviceaccount "kube-system/fsx-csi-controller-sa",
    } }
[#] building iamserviceaccount stack "eksctl-my-cluster-addon-iamserviceaccount-kube-system-fsx-csi-controller-sa"
[#] deploying stack "eksctl-my-cluster-addon-iamserviceaccount-kube-system-fsx-csi-controller-sa"
[#] waiting for CloudFormation stack "eksctl-my-cluster-addon-iamserviceaccount-kube-system-fsx-csi-controller-sa"
[#] created serviceaccount "kube-system/fsx-csi-controller-sa"
```

Perhatikan nama AWS CloudFormation tumpukan yang digunakan. Dalam contoh output sebelumnya, tumpukan diberi nama `eksctl-my-cluster-addon-iamserviceaccount-kube-system-fsx-csi-controller-sa`.

Sekarang setelah Anda membuat peran IAM driver Amazon FSx CSI, Anda dapat melanjutkan ke bagian berikutnya. Saat Anda menerapkan add-on dengan peran IAM ini, ia membuat dan dikonfigurasi untuk menggunakan akun layanan yang diberi nama `fsx-csi-controller-sa`. Akun layanan terikat pada Kubernetes yang diberi izin Kubernetes `clusterrole` yang diperlukan.

Langkah 2: Instal driver Amazon FSx CSI

Kami menyarankan Anda menginstal driver Amazon FSx CSI melalui add-on Amazon EKS untuk meningkatkan keamanan dan mengurangi jumlah pekerjaan. Untuk menambahkan add-on Amazon EKS ke cluster Anda, lihat [the section called “Buat add-on”](#). Untuk informasi selengkapnya tentang add-on, lihat [the section called “Add-on Amazon EKS”](#).

Important

Instalasi driver Amazon FSx CSI yang sudah ada sebelumnya di cluster dapat menyebabkan kegagalan instalasi add-on. Saat Anda mencoba menginstal versi add-on Amazon EKS saat Driver FSx CSI non-EKS ada, penginstalan akan gagal karena konflik sumber daya. Gunakan `OVERWRITE` bendera selama instalasi untuk mengatasi masalah ini.


```
aws eks create-addon --addon-name aws-fsx-csi-driver --cluster-name my-cluster  
--resolve-conflicts OVERWRITE
```

Atau, jika Anda menginginkan penginstalan driver Amazon FSx CSI yang dikelola sendiri, lihat [Instalasi](#) aktif. GitHub

Langkah 3: Menerapkan kelas penyimpanan, klaim volume persisten, dan aplikasi sampel

Prosedur ini menggunakan GitHub repositori [driver FSx for Lustre Container Storage Interface \(CSI\)](#) untuk menggunakan volume Lustre yang disediakan secara dinamis. FSx

1. Perhatikan grup keamanan untuk klaster Anda. Anda dapat melihatnya di AWS Management Console bawah bagian Jaringan atau dengan menggunakan perintah AWS CLI berikut. Ganti `my-cluster` dengan nama cluster yang ingin Anda gunakan.

```
aws eks describe-cluster --name my-cluster --query  
cluster.resourcesVpcConfig.clusterSecurityGroupId
```

2. Buat grup keamanan untuk sistem FSx file Amazon Anda sesuai dengan kriteria yang ditampilkan di [Grup Keamanan VPC Amazon](#) di Panduan Pengguna Amazon FSx untuk Lustre. Untuk VPC, pilih VPC cluster Anda seperti yang ditunjukkan di bawah bagian Networking. Untuk “grup keamanan yang terkait dengan klien Lustre Anda”, gunakan grup keamanan klaster Anda. Anda dapat meninggalkan aturan keluar sendiri untuk memungkinkan Semua lalu lintas.
3. Unduh manifes kelas penyimpanan dengan perintah berikut.

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-fsx-csi-driver/master/  
examples/kubernetes/dynamic_provisioning/specs/storageclass.yaml
```

4. Edit bagian parameter `storageclass.yaml` file. Ganti setiap nilai contoh dengan nilai Anda sendiri.

```
parameters:  
  subnetId: subnet-0eabfaa81fb22bcaf  
  securityGroupIds: sg-068000ccf82dfba88  
  deploymentType: PERSISTENT_1  
  automaticBackupRetentionDays: "1"
```

```
dailyAutomaticBackupStartTime: "00:00"  
copyTagsToBackups: "true"  
perUnitStorageThroughput: "200"  
dataCompressionType: "NONE"  
weeklyMaintenanceStartTime: "7:09:00"  
fileSystemTypeVersion: "2.12"
```

- **subnetId**— ID subnet tempat sistem file Amazon FSx untuk Lustre harus dibuat. Amazon FSx for Lustre tidak didukung di semua Availability Zone. Buka konsol Amazon FSx for Lustre di <https://console.aws.amazon.com/fsx/> untuk mengonfirmasi bahwa subnet yang ingin Anda gunakan berada di Availability Zone yang didukung. Subnet dapat menyertakan node Anda, atau dapat berupa subnet atau VPC yang berbeda:
 - Anda dapat memeriksa subnet node di AWS Management Console dengan memilih grup node di bawah bagian Compute.
 - Jika subnet yang Anda tentukan bukan subnet yang sama dengan yang Anda miliki node, maka Anda VPCs harus [terhubung](#), dan Anda harus memastikan bahwa Anda memiliki port yang diperlukan terbuka di grup keamanan Anda.
- **securityGroupIds**— ID grup keamanan yang Anda buat untuk sistem file.
- **deploymentType**(opsional) — Jenis penyebaran sistem file. Nilai yang valid adalah SCRATCH_1, SCRATCH_2, PERSISTENT_1, dan PERSISTENT_2. Untuk informasi selengkapnya tentang jenis penerapan, lihat [Membuat sistem file Amazon FSx untuk Lustre](#).
- parameter lain (opsional) — Untuk informasi tentang parameter lainnya, lihat [StorageClassMenedit](#) GitHub.

5. Buat manifes kelas penyimpanan.

```
kubectl apply -f storageclass.yaml
```

Contoh output adalah sebagai berikut.

```
storageclass.storage.k8s.io/fsx-sc created
```

6. Unduh manifes klaim volume persisten.

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-fsx-csi-driver/master/  
examples/kubernetes/dynamic_provisioning/specs/claim.yaml
```

7. (Opsional) Edit file `claim.yaml`. Ubah `1200Gi` ke salah satu nilai kenaikan berikut, berdasarkan kebutuhan penyimpanan Anda dan `deploymentType` yang Anda pilih pada langkah sebelumnya.

```
storage: 1200Gi
```

- `SCRATCH_2` dan `PERSISTENT` — `1.2 TiB`, `2.4 TiB`, atau kenaikan `2,4 TiB` lebih dari `2,4 TiB`.
- `SCRATCH_1` — `1.2 TiB`, `2.4 TiB`, `3.6 TiB`, atau kenaikan `3,6 TiB` lebih dari `3,6 TiB`.

8. Buat klaim volume persisten.

```
kubectl apply -f claim.yaml
```

Contoh output adalah sebagai berikut.

```
persistentvolumeclaim/fsx-claim created
```

9. Konfirmasikan bahwa sistem file disediakan.

```
kubectl describe pvc
```

Contoh output adalah sebagai berikut.

```
Name:          fsx-claim
Namespace:     default
StorageClass:  fsx-sc
Status:        Bound
[...]
```

 Note

Status mungkin ditampilkan sebagai `Pending` selama 5-10 menit, sebelum berubah ke `Bound`. Jangan lanjutkan dengan langkah berikutnya sampai Status ada `Bound`. Jika Status ditampilkan `Pending` selama lebih dari 10 menit, gunakan pesan peringatan di `Events` sebagai referensi untuk mengatasi masalah apa pun.

10. Deploy aplikasi sampel.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-fsx-csi-driver/master/examples/kubernetes/dynamic_provisioning/specs/pod.yaml
```

11. Verifikasi bahwa aplikasi sampel sedang dijalankan.

```
kubectl get pods
```

Contoh output adalah sebagai berikut.

NAME	READY	STATUS	RESTARTS	AGE
fsx-app	1/1	Running	0	8s

12. Verifikasi bahwa sistem file dipasang dengan benar oleh aplikasi.

```
kubectl exec -ti fsx-app -- df -h
```

Contoh output adalah sebagai berikut.

Filesystem	Size	Used	Avail	Use%	Mounted on
overlay	80G	4.0G	77G	5%	/
tmpfs	64M	0	64M	0%	/dev
tmpfs	3.8G	0	3.8G	0%	/sys/fs/cgroup
192.0.2.0@tcp:/abcdef01	1.1T	7.8M	1.1T	1%	/data
/dev/nvme0n1p1	80G	4.0G	77G	5%	/etc/hosts
shm	64M	0	64M	0%	/dev/shm
tmpfs	6.9G	12K	6.9G	1%	/run/secrets/kubernetes.io/
serviceaccount					
tmpfs	3.8G	0	3.8G	0%	/proc/acpi
tmpfs	3.8G	0	3.8G	0%	/sys/firmware

13. Verifikasi bahwa data telah ditulis ke sistem file FSx for Lustre oleh aplikasi sampel.

```
kubectl exec -it fsx-app -- ls /data
```

Contoh output adalah sebagai berikut.

```
out.txt
```

Contoh keluaran ini menunjukkan bahwa aplikasi sampel berhasil menulis out .txt file ke sistem file.

 Note

Sebelum menghapus cluster, pastikan untuk menghapus sistem file FSx for Lustre. Untuk informasi selengkapnya, lihat [Membersihkan sumber daya](#) di Panduan Pengguna FSx untuk Lustre.

Tuning kinerja FSx untuk Lustre

Saat menggunakan Lustre dengan Amazon EKS, Anda dapat mengoptimalkan kinerja dengan menerapkan penyetelan Lustre FSx selama inisialisasi node. Pendekatan yang disarankan adalah menggunakan data pengguna template peluncuran untuk memastikan konfigurasi yang konsisten di semua node.

Penyetelan ini meliputi:

- Pengoptimalan jaringan dan RPC
- Manajemen modul lustre
- Penyetelan LRU (Unit Sumber Daya Kunci)
- Pengaturan kontrol cache klien
- Kontrol RPC untuk OST dan MDC

Untuk petunjuk rinci tentang penerapan tuning kinerja ini:

- Untuk mengoptimalkan kinerja untuk node standar (non-EFA), lihat [the section called “Optimalkan \(Non-EFA\)”](#) skrip lengkap yang dapat ditambahkan ke data pengguna template peluncuran Anda.
- Untuk mengoptimalkan kinerja node yang mendukung EFA, lihat. [the section called “Optimalkan \(EFA\)”](#)

Optimalkan Amazon FSx untuk kinerja Lustre pada node (EFA)

Topik ini menjelaskan cara mengatur penyetelan Elastic Fabric Adapter (EFA) dengan Amazon EKS dan Amazon untuk Lustre. FSx

Note

- Untuk informasi tentang membuat dan menerapkan driver FSx untuk Lustre CSI, lihat. [the section called “Menyebarkan driver”](#)
- Untuk mengoptimalkan node standar tanpa EFA, lihat. [the section called “Optimalkan \(Non-EFA\)”](#)

Langkah 1. Buat kluster EKS

Buat cluster menggunakan file konfigurasi yang disediakan:

```
# Create cluster using efa-cluster.yaml
eksctl create cluster -f efa-cluster.yaml
```

Contoh efa-cluster.yaml:

```
#efa-cluster.yaml

apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: csi-fsx
  region: us-east-1
  version: "1.30"

iam:
  withOIDC: true

availabilityZones: ["us-east-1a", "us-east-1d"]

managedNodeGroups:
  - name: my-efa-ng
    instanceType: c6gn.16xlarge
    minSize: 1
```

```

desiredCapacity: 1
maxSize: 1
availabilityZones: ["us-east-1b"]
volumeSize: 300
privateNetworking: true
amiFamily: Ubuntu2204
efaEnabled: true
preBootstrapCommands:
  - |
    #!/bin/bash
    eth_intf="$(/sbin/ip -br -4 a sh | grep $(hostname -i)/ | awk '{print $1}')"
    efa_version=$(modinfo efa | awk '/^version:/ {print $2}' | sed 's/[^0-9.]//g')
    min_efa_version="2.12.1"

    if [[ "$(printf '%s\n' "$min_efa_version" "$efa_version" | sort -V | head -
n1)" != "$min_efa_version" ]]; then
        sudo curl -O https://efa-installer.amazonaws.com/aws-efa-
installer-1.37.0.tar.gz
        tar -xf aws-efa-installer-1.37.0.tar.gz && cd aws-efa-installer
        echo "Installing EFA driver"
        sudo apt-get update && apt-get upgrade -y
        sudo apt install -y pciutils environment-modules libnl-3-dev libnl-
route-3-200 libnl-route-3-dev dkms
        sudo ./efa_installer.sh -y
        modinfo efa
    else
        echo "Using EFA driver version $efa_version"
    fi

    echo "Installing Lustre client"
    sudo wget -O - https://fsx-lustre-client-repo-public-keys.s3.amazonaws.com/fsx-
ubuntu-public-key.asc | gpg --dearmor | sudo tee /usr/share/keyrings/fsx-ubuntu-public-
key.gpg > /dev/null
    sudo echo "deb [signed-by=/usr/share/keyrings/fsx-ubuntu-public-key.gpg]
https://fsx-lustre-client-repo.s3.amazonaws.com/ubuntu jammy main" > /etc/apt/
sources.list.d/fsxlustreclientrepo.list
    sudo apt update | tail
    sudo apt install -y lustre-client-modules-$(uname -r) amazon-ec2-utils | tail
    modinfo lustre

    echo "Loading Lustre/EFA modules..."
    sudo /sbin/modprobe lnet
    sudo /sbin/modprobe kefalnd ipif_name="$eth_intf"
    sudo /sbin/modprobe ksocklnd

```

```

sudo lnctl net configure

echo "Configuring TCP interface..."
sudo lnctl net del --net tcp 2> /dev/null
sudo lnctl net add --net tcp --if $eth_intf

# For P5 instance type which supports 32 network cards,
# by default add 8 EFA interfaces selecting every 4th device (1 per PCI bus)
echo "Configuring EFA interface(s)..."
instance_type="$(ec2-metadata --instance-type | awk '{ print $2 }')"
num_efa_devices="$(ls -1 /sys/class/infiniband | wc -l)"
echo "Found $num_efa_devices available EFA device(s)"

if [[ "$instance_type" == "p5.48xlarge" || "$instance_type" ==
"p5e.48xlarge" ]]; then
    for intf in $(ls -1 /sys/class/infiniband | awk 'NR % 4 == 1'); do
        sudo lnctl net add --net efa --if $intf --peer-credits 32
    done
else
    # Other instances: Configure 2 EFA interfaces by default if the instance
    supports multiple network cards,
    # or 1 interface for single network card instances
    # Can be modified to add more interfaces if instance type supports it
    sudo lnctl net add --net efa --if $(ls -1 /sys/class/infiniband | head -
n1) --peer-credits 32
    if [[ $num_efa_devices -gt 1 ]]; then
        sudo lnctl net add --net efa --if $(ls -1 /sys/class/infiniband | tail
-n1) --peer-credits 32
    fi
fi

echo "Setting discovery and UDSP rule"
sudo lnctl set discovery 1
sudo lnctl udsp add --src efa --priority 0
sudo /sbin/modprobe lustre

sudo lnctl net show
echo "Added $(sudo lnctl net show | grep -c '@efa') EFA interface(s)"

```

Langkah 2. Buat grup simpul

Buat grup simpul yang mendukung EFA:


```
# Create node group using efa-ng.yaml
eksctl create nodegroup -f efa-ng.yaml
```

Important

=== Sesuaikan nilai-nilai ini untuk lingkungan Anda di bagian# 5. Mount FSx filesystem.

```
FSX_DNS="<your-fsx-filesystem-dns>" # Needs to be adjusted.
MOUNT_NAME="<your-mount-name>" # Needs to be adjusted.
MOUNT_POINT="</your/mount/point>" # Needs to be adjusted.
```

===

Contohefa-ng.yaml:

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: final-efa
  region: us-east-1

managedNodeGroups:
  - name: ng-1
    instanceType: c6gn.16xlarge
    minSize: 1
    desiredCapacity: 1
    maxSize: 1
    availabilityZones: ["us-east-1a"]
    volumeSize: 300
    privateNetworking: true
    amiFamily: Ubuntu2204
    efaEnabled: true
    preBootstrapCommands:
      - |
        #!/bin/bash
        exec 1> >(logger -s -t $(basename $0)) 2>&1
```

```
#####
#                                     Configuration Section
#
#####

# File System Configuration
FSX_DNS="<your-fsx-file-system-dns>" # Needs to be adjusted.
MOUNT_NAME="<your-mount-name>" # Needs to be adjusted.
MOUNT_POINT="</your-mount/point>" # Needs to be adjusted.

# Lustre Tuning Parameters
LUSTRE_LRU_MAX_AGE=600000
LUSTRE_MAX_CACHED_MB=64
LUSTRE_OST_MAX_RPC=32
LUSTRE_MDC_MAX_RPC=64
LUSTRE_MDC_MOD_RPC=50

# File paths
FUNCTIONS_SCRIPT="/usr/local/bin/lustre_functions.sh"
TUNINGS_SCRIPT="/usr/local/bin/apply_lustre_tunings.sh"
SERVICE_FILE="/etc/systemd/system/lustre-tunings.service"

#EFA
MIN_EFA_VERSION="2.12.1"

# Function to check if a command was successful
check_success() {
    if [ $? -eq 0 ]; then
        echo "SUCCESS: $1"
    else
        echo "FAILED: $1"
        return 1
    fi
}

echo "*****Starting FSx for Lustre configuration*****"

# 1. Install Lustre client
if grep -q '^ID=ubuntu' /etc/os-release; then
    echo "Detected Ubuntu, proceeding with Lustre setup..."
    # Add Lustre repository
```

```

        sudo wget -O - https://fsx-lustre-client-repo-public-keys.s3.amazonaws.com/
fsx-ubuntu-public-key.asc | sudo gpg --dearmor | sudo tee /usr/share/keyrings/fsx-
ubuntu-public-key.gpg > /dev/null

        echo "deb [signed-by=/usr/share/keyrings/fsx-ubuntu-public-key.gpg]
https://fsx-lustre-client-repo.s3.amazonaws.com/ubuntu jammy main" | sudo tee /etc/
apt/sources.list.d/fsxlustreclientrepo.list

        sudo apt-get update
        sudo apt-get install -y lustre-client-modules-$(uname -r)
        sudo apt-get install -y lustre-client
    else
        echo "Not Ubuntu, exiting"
        exit 1
    fi

    check_success "Install Lustre client"

    # Ensure Lustre tools are in the PATH
    export PATH=$PATH:/usr/sbin

    # 2. Apply network and RPC tunings
    echo "*****Applying network and RPC tunings*****"
    if ! grep -q "options ptlrpc ptlrpcd_per_cpt_max" /etc/modprobe.d/
modprobe.conf; then
        echo "options ptlrpc ptlrpcd_per_cpt_max=64" | sudo tee -a /etc/modprobe.d/
modprobe.conf
        check_success "Set ptlrpcd_per_cpt_max"
    else
        echo "ptlrpcd_per_cpt_max already set in modprobe.conf"
    fi

    if ! grep -q "options ksocklnd credits" /etc/modprobe.d/modprobe.conf; then
        echo "options ksocklnd credits=2560" | sudo tee -a /etc/modprobe.d/
modprobe.conf
        check_success "Set ksocklnd credits"
    else
        echo "ksocklnd credits already set in modprobe.conf"
    fi

    # 3. Load Lustre modules
    manage_lustre_modules() {
        echo "Checking for existing Lustre modules..."
        if lsmod | grep -q lustre; then

```

```

        echo "Existing Lustre modules found."

        # Check for mounted Lustre filesystems
        echo "Checking for mounted Lustre filesystems..."
        if mount | grep -q "type lustre"; then
            echo "Found mounted Lustre filesystems. Attempting to unmount..."
            mounted_fs=$(mount | grep "type lustre" | awk '{print $3}')
            for fs in $mounted_fs; do
                echo "Unmounting $fs"
                sudo umount $fs
                check_success "Unmount filesystem $fs"
            done
        else
            echo "No Lustre filesystems mounted."
        fi

        # After unmounting, try to remove modules
        echo "Attempting to remove Lustre modules..."
        sudo lustre_rmmod
        if [ $? -eq 0 ]; then
            echo "SUCCESS: Removed existing Lustre modules"
        else
            echo "WARNING: Could not remove Lustre modules. They may still be
in use."

            echo "Please check for any remaining Lustre processes or mounts."
            return 1
        fi
    else
        echo "No existing Lustre modules found."
    fi

    echo "Loading Lustre modules..."
    sudo modprobe lustre
    check_success "Load Lustre modules" || exit 1

    echo "Checking loaded Lustre modules:"
    lsmod | grep lustre
}

# Managing Lustre kernel modules
echo "*****Managing Lustre kernel modules*****"
manage_lustre_modules

# 4. Initializing Lustre networking

```

```

echo "*****Initializing Lustre networking*****"
sudo lctl network up
check_success "Initialize Lustre networking" || exit 1

# 4.5 EFA Setup and Configuration
setup_efa() {
    echo "*****Starting EFA Setup*****"

    # Get interface and version information
    eth_intf="$(/sbin/ip -br -4 a sh | grep $(hostname -i)/ | awk '{print
$1}')"

    efa_version=$(modinfo efa | awk '/^version:/ {print $2}' | sed 's/[^0-9.]//
g')

    min_efa_version=$MIN_EFA_VERSION

    # Install or verify EFA driver
    if [[ "$(printf '%s\n' "$min_efa_version" "$efa_version" | sort -V | head -
n1)" != "$min_efa_version" ]]; then
        echo "Installing EFA driver..."
        sudo curl -O https://efa-installer.amazonaws.com/aws-efa-
installer-1.37.0.tar.gz
        tar -xf aws-efa-installer-1.37.0.tar.gz && cd aws-efa-installer

        # Install dependencies
        sudo apt-get update && apt-get upgrade -y
        sudo apt install -y pciutils environment-modules libnl-3-dev libnl-
route-3-200 libnl-route-3-dev dkms

        # Install EFA
        sudo ./efa_installer.sh -y
        modinfo efa
    else
        echo "Using existing EFA driver version $efa_version"
    fi
}

configure_efa_network() {
    echo "*****Configuring EFA Network*****"

    # Load required modules
    echo "Loading network modules..."
    sudo /sbin/modprobe lnet
    sudo /sbin/modprobe kefalnd ipif_name="$eth_intf"
    sudo /sbin/modprobe ksocklnd

```

```

# Initialize LNet
echo "Initializing LNet..."
sudo lnctl lnet configure

# Configure TCP interface
echo "Configuring TCP interface..."
sudo lnctl net del --net tcp 2> /dev/null
sudo lnctl net add --net tcp --if $eth_intf

# For P5 instance type which supports 32 network cards,
# by default add 8 EFA interfaces selecting every 4th device (1 per PCI
bus)

echo "Configuring EFA interface(s)..."
instance_type="$(ec2-metadata --instance-type | awk '{ print $2 }')"
num_efa_devices="$(ls -1 /sys/class/infiniband | wc -l)"
echo "Found $num_efa_devices available EFA device(s)"

if [[ "$instance_type" == "p5.48xlarge" || "$instance_type" ==
"p5e.48xlarge" ]]; then
    # P5 instance configuration
    for intf in $(ls -1 /sys/class/infiniband | awk 'NR % 4 == 1'); do
        sudo lnctl net add --net efa --if $intf --peer-credits 32
    done
else
    # Standard configuration
    # Other instances: Configure 2 EFA interfaces by default if the
instance supports multiple network cards,
    # or 1 interface for single network card instances
    # Can be modified to add more interfaces if instance type supports it
    sudo lnctl net add --net efa --if $(ls -1 /sys/class/infiniband |
head -n1) --peer-credits 32
    if [[ $num_efa_devices -gt 1 ]]; then
        sudo lnctl net add --net efa --if $(ls -1 /sys/class/infiniband |
tail -n1) --peer-credits 32
    fi
fi

# Configure discovery and UDSP
echo "Setting up discovery and UDSP rules..."
sudo lnctl set discovery 1
sudo lnctl udsp add --src efa --priority 0
sudo /sbin/modprobe lustre

```

```
# Verify configuration
echo "Verifying EFA network configuration..."
sudo lnctl net show
echo "Added $(sudo lnctl net show | grep -c '@efa') EFA interface(s)"
}

# Main execution
setup_efa
configure_efa_network

# 5. Mount FSx filesystem
if [ ! -z "$FSX_DNS" ] && [ ! -z "$MOUNT_NAME" ]; then
    echo "*****Creating mount point*****"
    sudo mkdir -p $MOUNT_POINT
    check_success "Create mount point"

    echo "Mounting FSx filesystem..."
    sudo mount -t lustre ${FSX_DNS}@tcp:${MOUNT_NAME} ${MOUNT_POINT}
    check_success "Mount FSx filesystem"
else
    echo "Skipping FSx mount as DNS or mount name is not provided"
fi

# 6. Applying Lustre performance tunings
echo "*****Applying Lustre performance tunings*****"

# Get number of CPUs
NUM_CPUS=$(nproc)

# Calculate LRU size (100 * number of CPUs)
LRU_SIZE=$((100 * NUM_CPUS))

#Apply LRU tunings
echo "Apply LRU tunings"
sudo lctl set_param ldlm.namespaces.*.lru_max_age=${LUSTRE_LRU_MAX_AGE}
check_success "Set lru_max_age"
sudo lctl set_param ldlm.namespaces.*.lru_size=$LRU_SIZE
check_success "Set lru_size"

# Client Cache Control
sudo lctl set_param llite.*.max_cached_mb=${LUSTRE_MAX_CACHED_MB}
check_success "Set max_cached_mb"

# RPC Controls
```

```

sudo lctl set_param osc.*OST*.max_rpcs_in_flight=${LUSTRE_OST_MAX_RPC}
check_success "Set OST max_rpcs_in_flight"

sudo lctl set_param mdc.*.max_rpcs_in_flight=${LUSTRE_MDC_MAX_RPC}
check_success "Set MDC max_rpcs_in_flight"

sudo lctl set_param mdc.*.max_mod_rpcs_in_flight=${LUSTRE_MDC_MOD_RPC}
check_success "Set MDC max_mod_rpcs_in_flight"

# 7. Verify all tunings
echo "*****Verifying all tunings*****"

# Function to verify parameter value
verify_param() {
    local param=$1
    local expected=$2
    local actual=$3

    if [ "$actual" == "$expected" ]; then
        echo "SUCCESS: $param is correctly set to $expected"
    else
        echo "WARNING: $param is set to $actual (expected $expected)"
    fi
}

echo "Verifying all parameters:"

# LRU tunings
actual_lru_max_age=$(lctl get_param -n ldlm.namespaces.*.lru_max_age | head -1)
verify_param "lru_max_age" "600000" "$actual_lru_max_age"

actual_lru_size=$(lctl get_param -n ldlm.namespaces.*.lru_size | head -1)
verify_param "lru_size" "$LRU_SIZE" "$actual_lru_size"

# Client Cache
actual_max_cached_mb=$(lctl get_param -n llite.*.max_cached_mb | grep
"max_cached_mb:" | awk '{print $2}')
verify_param "max_cached_mb" "64" "$actual_max_cached_mb"

# RPC Controls
actual_ost_rpcs=$(lctl get_param -n osc.*OST*.max_rpcs_in_flight | head -1)
verify_param "OST max_rpcs_in_flight" "32" "$actual_ost_rpcs"

actual_mdc_rpcs=$(lctl get_param -n mdc.*.max_rpcs_in_flight | head -1)

```



```

verify_param "MDC max_rpcs_in_flight" "64" "$actual_mdc_rpcs"

actual_mdc_mod_rpcs=$(lctl get_param -n mdc.*.max_mod_rpcs_in_flight | head -1)
verify_param "MDC max_mod_rpcs_in_flight" "50" "$actual_mdc_mod_rpcs"

# Network and RPC configurations from modprobe.conf
actual_ptlrpc=$(grep "ptlrpc ptlrpd_per_cpt_max" /etc/modprobe.d/modprobe.conf
| awk '{print $3}')
verify_param "ptlrpd_per_cpt_max" "ptlrpd_per_cpt_max=64" "$actual_ptlrpc"

actual_ksocklnd=$(grep "ksocklnd credits" /etc/modprobe.d/modprobe.conf | awk
'{print $3}')
verify_param "ksocklnd credits" "credits=2560" "$actual_ksocklnd"

# 8. Setup persistence
setup_persistence() {
    # Create functions file
    cat << EOF > $FUNCTIONS_SCRIPT
#!/bin/bash

apply_lustre_tunings() {
    local NUM_CPUS=$(nproc)
    local LRU_SIZE=$((100 * NUM_CPUS))

    echo "Applying Lustre performance tunings..."
    lctl set_param ldln.namespaces.*.lru_max_age=$LUSTRE_LRU_MAX_AGE
    lctl set_param ldln.namespaces.*.lru_size=$LRU_SIZE
    lctl set_param llite.*.max_cached_mb=$LUSTRE_MAX_CACHED_MB
    lctl set_param osc.*OST*.max_rpcs_in_flight=$LUSTRE_OST_MAX_RPC
    lctl set_param mdc.*.max_rpcs_in_flight=$LUSTRE_MDC_MAX_RPC
    lctl set_param mdc.*.max_mod_rpcs_in_flight=$LUSTRE_MDC_MOD_RPC
}
EOF

    # Create tuning script
    cat << EOF > $TUNINGS_SCRIPT
#!/bin/bash
exec 1> >(logger -s -t $(basename $0)) 2>&1

source $FUNCTIONS_SCRIPT

# Function to check if Lustre is mounted
is_lustre_mounted() {
    mount | grep -q "type lustre"

```

```

}

# Function to mount Lustre
mount_lustre() {
    echo "Mounting Lustre filesystem..."
    mkdir -p $MOUNT_POINT
    mount -t lustre ${FSX_DNS}@tcp:${MOUNT_NAME} $MOUNT_POINT
    return $?
}

# Main execution
# Try to mount if not already mounted
if ! is_lustre_mounted; then
    echo "Lustre filesystem not mounted, attempting to mount..."
    mount_lustre
fi

# Wait for successful mount (up to 5 minutes)
for i in {1..30}; do
    if is_lustre_mounted; then
        echo "Lustre filesystem mounted, applying tunings..."
        apply_lustre_tunings
        exit 0
    fi
    echo "Waiting for Lustre filesystem to be mounted... (attempt $i/30)"
    sleep 10
done

echo "Timeout waiting for Lustre filesystem mount"
exit 1
EOF

# Create systemd service

# Create systemd directory if it doesn't exist
sudo mkdir -p /etc/systemd/system/

    # Create service file directly for Ubuntu
    cat << EOF > $SERVICE_FILE
[Unit]
Description=Apply Lustre Performance Tunings
After=network.target remote-fs.target

[Service]

```

```
Type=oneshot
ExecStart=/bin/bash -c 'source $FUNCTIONS_SCRIPT && $TUNINGS_SCRIPT'
RemainAfterExit=yes

[Install]
WantedBy=multi-user.target
EOF

# Make scripts executable and enable service
sudo chmod +x $FUNCTIONS_SCRIPT
sudo chmod +x $TUNINGS_SCRIPT
systemctl enable lustre-tunings.service
systemctl start lustre-tunings.service
}

echo "*****Setting up persistent tuning*****"
setup_persistence

echo "FSx for Lustre configuration completed."
```

(Opsional) Langkah 3. Verifikasi pengaturan EFA

SSH menjadi simpul:

```
# Get instance ID from EKS console or AWS CLI
ssh -i /path/to/your-key.pem ec2-user@<node-internal-ip>
```

Verifikasi konfigurasi EFA:

```
sudo lnctl net show
```

Periksa log pengaturan:

```
sudo cat /var/log/cloud-init-output.log
```

Berikut contoh output yang diharapkan untuk `lnctl net show`:

```
net:
- net type: tcp
...
- net type: efa
```

```
local NI(s):  
  - nid: xxx.xxx.xxx.xxx@efa  
  status: up
```

Contoh penerapan

sebuah. Buat claim.yaml

```
#claim.yaml  
  
apiVersion: v1  
kind: PersistentVolumeClaim  
metadata:  
  name: fsx-claim-efa  
spec:  
  accessModes:  
    - ReadWriteMany  
  storageClassName: ""  
  resources:  
    requests:  
      storage: 4800Gi  
  volumeName: fsx-pv
```

Terapkan klaim:

```
kubectl apply -f claim.yaml
```

b. Buat pv.yaml

Perbarui<replaceable-placeholders>:

```
#pv.yaml  
  
apiVersion: v1  
kind: PersistentVolume  
metadata:  
  name: fsx-pv  
spec:  
  capacity:  
    storage: 4800Gi  
  volumeMode: Filesystem  
  accessModes:  
    - ReadWriteMany
```

```
mountOptions:
  - flock
persistentVolumeReclaimPolicy: Recycle
csi:
  driver: fsx.csi.aws.com
  volumeHandle: fs-<1234567890abcdef0>
  volumeAttributes:
    dnsname: fs-<1234567890abcdef0>.fsx.us-east-1.amazonaws.com
    mountname: <abcdef01>
```

Terapkan volume persisten:

```
kubectl apply -f pv.yaml
```

c. Buat pod.yaml

```
#pod.yaml

apiVersion: v1
kind: Pod
metadata:
  name: fsx-efa-app
spec:
  containers:
    - name: app
      image: amazonlinux:2
      command: ["/bin/sh"]
      args: ["-c", "while true; do dd if=/dev/urandom bs=100M count=20 > data/test_file;
sleep 10; done"]
      resources:
        requests:
          vpc.amazonaws.com/efa: 1
        limits:
          vpc.amazonaws.com/efa: 1
      volumeMounts:
        - name: persistent-storage
          mountPath: /data
  volumes:
    - name: persistent-storage
      persistentVolumeClaim:
        claimName: fsx-claim-efa
```

Terapkan Pod:

```
kubectl apply -f pod.yaml
```

Perintah verifikasi tambahan

Verifikasi pemasangan Pod dan tulis ke sistem file:

```
kubectl exec -ti fsx-efa-app -- df -h | grep data
# Expected output:
# <192.0.2.0>@tcp:/<abcdef01> 4.5T 1.2G 4.5T 1% /data

kubectl exec -ti fsx-efa-app -- ls /data
# Expected output:
# test_file
```

SSH ke node untuk memverifikasi lalu lintas melewati EFA:

```
sudo lnctl net show -v
```

Output yang diharapkan akan menunjukkan antarmuka EFA dengan statistik lalu lintas.

Informasi terkait

- [the section called “Menyebarkan driver”](#)
- [the section called “Optimalkan \(Non-EFA\)”](#)
- [Amazon FSx untuk Kinerja Kilau](#)
- [Elastic Fabric Adapter](#)

Optimalkan Amazon FSx untuk kinerja Lustre pada node (non-EFA)

Anda dapat mengoptimalkan Amazon FSx untuk kinerja Lustre dengan menerapkan parameter penyetelan selama inisialisasi node menggunakan data pengguna template peluncuran.

Note

- Untuk informasi tentang membuat dan menerapkan driver FSx untuk Lustre CSI, lihat. [the section called “Menyebarkan driver”](#) Untuk mengoptimalkan kinerja dengan node yang mendukung EFA, lihat. [the section called “Optimalkan \(EFA\)”](#)

Mengapa menggunakan data pengguna template peluncuran?

- Menerapkan penyetelan secara otomatis selama inisialisasi node.
- Memastikan konfigurasi yang konsisten di semua node.
- Menghilangkan kebutuhan untuk konfigurasi node manual.

Contoh ikhtisar skrip

Contoh skrip yang didefinisikan dalam topik ini melakukan operasi ini:

1. Install Lustre client

- Secara otomatis mendeteksi versi OS Anda: Amazon Linux 2 (AL2) atau Amazon Linux 2023 (AL2023).
- Menginstal paket klien Lustre yang sesuai.

2. Apply network and RPC tunings

- Set `ptlrpcd_per_cpt_max=64` untuk pemrosesan RPC paralel.
- Mengkonfigurasi `ksocklnd credits=2560` untuk mengoptimalkan buffer jaringan.

3. Load Lustre modules

- Aman menghapus modul Lustre yang ada jika ada.
- Menangani pelepasan sistem file yang ada.
- Memuat modul Lustre segar.

4. Lustre Network Initialization

- Menginisialisasi konfigurasi jaringan Lustre.
- Menyiapkan parameter jaringan yang diperlukan.

5. Mount FSx filesystem

- Anda harus menyesuaikan nilai untuk lingkungan Anda di bagian ini.

6. Apply tunings

- Penyetelan LRU (Unit Sumber Daya Kunci):
 - `lru_max_age=600000`
 - `lru_size` dihitung berdasarkan jumlah CPU
- Kontrol Cache Klien: `max_cached_mb=64`
- Kontrol RPC:
 - OST `max_rpcs_in_flight=32`
 - MDC `max_rpcs_in_flight=64`
 - MDC `max_mod_rpcs_in_flight=50`

7. Verify tunings

- Memverifikasi semua tuning yang diterapkan.
- Melaporkan keberhasilan atau peringatan untuk setiap parameter.

8. Setup persistence

- Anda harus menyesuaikan nilai untuk lingkungan Anda di bagian ini juga.
- Secara otomatis mendeteksi versi OS Anda (AL2 atau AL2 023) untuk menentukan Systemd layanan mana yang akan diterapkan.
- Sistem dimulai.
- Systemd memulai `lustre-tunings` layanan (karena `WantedBy=multi-user.target`).
- Layanan berjalan `apply_lustre_tunings.sh` yang:
 - Memeriksa apakah sistem file sudah terpasang.
 - Pasang sistem file jika tidak dipasang.
 - Menunggu pemasangan yang berhasil (hingga lima menit).
 - Menerapkan parameter penyetelan setelah pemasangan berhasil.
- Pengaturan tetap aktif hingga reboot.
- Layanan keluar setelah skrip selesai.
 - Systemd menandai layanan sebagai “aktif (keluar)”.
- Proses berulang pada reboot berikutnya.

Membuat templat peluncuran

1. Buka EC2 konsol Amazon di <https://console.aws.amazon.com/ec2/>.
2. Pilih Luncurkan Template.
3. Pilih Buat templat peluncuran.
4. Di Detail lanjutan, cari bagian Data pengguna.
5. Tempel skrip di bawah ini, perbarui apa pun sesuai kebutuhan.

Important

Sesuaikan nilai-nilai ini untuk lingkungan Anda di kedua bagian # 5. Mount FSx filesystem dan `setup_persistence()` fungsi `apply_lustre_tunings.sh` di bagian# 8. Setup persistence:

```
FSX_DNS="<your-fsx-filesystem-dns>" # Needs to be adjusted.  
MOUNT_NAME="<your-mount-name>" # Needs to be adjusted.  
MOUNT_POINT="</your/mount/point>" # Needs to be adjusted.
```

```
MIME-Version: 1.0  
Content-Type: multipart/mixed; boundary=="MYBOUNDARY=="  
--MYBOUNDARY--  
Content-Type: text/x-shellscript; charset="us-ascii"  
#!/bin/bash  
exec 1> >(logger -s -t $(basename $0)) 2>&1  
# Function definitions  
check_success() {  
    if [ $? -eq 0 ]; then  
        echo "SUCCESS: $1"  
    else  
        echo "FAILED: $1"  
        return 1  
    fi  
}  
apply_tunings() {  
    local NUM_CPUS=$(nproc)  
    local LRU_SIZE=$((100 * NUM_CPUS))  
    local params=(  
        "ldlm.namespaces.*.lru_max_age=600000"  
    )  
}
```

```

        "ldlm.namespaces.*.lru_size=$LRU_SIZE"
        "llite.*.max_cached_mb=64"
        "osc.*OST*.max_rpcs_in_flight=32"
        "mdc.*.max_rpcs_in_flight=64"
        "mdc.*.max_mod_rpcs_in_flight=50"
    )
    for param in "${params[@]"; do
        lctl set_param $param
        check_success "Set ${param%*=*}"
    done
}

verify_param() {
    local param=$1
    local expected=$2
    local actual=$3

    if [ "$actual" == "$expected" ]; then
        echo "SUCCESS: $param is correctly set to $expected"
    else
        echo "WARNING: $param is set to $actual (expected $expected)"
    fi
}

verify_tunings() {
    local NUM_CPUS=$(nproc)
    local LRU_SIZE=$((100 * NUM_CPUS))
    local params=(
        "ldlm.namespaces.*.lru_max_age:600000"
        "ldlm.namespaces.*.lru_size:$LRU_SIZE"
        "llite.*.max_cached_mb:64"
        "osc.*OST*.max_rpcs_in_flight:32"
        "mdc.*.max_rpcs_in_flight:64"
        "mdc.*.max_mod_rpcs_in_flight:50"
    )
    echo "Verifying all parameters:"
    for param in "${params[@]"; do
        name="${param%*:*}"
        expected="${param#*:}"
        actual=$(lctl get_param -n $name | head -1)
        verify_param "${name##*.*}" "$expected" "$actual"
    done
}

setup_persistence() {
    # Create functions file
    cat << 'EOF' > /usr/local/bin/lustre_functions.sh

```

```
#!/bin/bash
apply_lustre_tunings() {
    local NUM_CPUS=$(nproc)
    local LRU_SIZE=$((100 * NUM_CPUS))

    echo "Applying Lustre performance tunings..."
    lctl set_param ldlm.namespaces.*.lru_max_age=600000
    lctl set_param ldlm.namespaces.*.lru_size=$LRU_SIZE
    lctl set_param llite.*.max_cached_mb=64
    lctl set_param osc.*OST*.max_rpcs_in_flight=32
    lctl set_param mdc.*.max_rpcs_in_flight=64
    lctl set_param mdc.*.max_mod_rpcs_in_flight=50
}
EOF

# Create tuning script
cat << 'EOF' > /usr/local/bin/apply_lustre_tunings.sh
#!/bin/bash
exec 1> >(logger -s -t $(basename $0)) 2>&1
# Source the functions
source /usr/local/bin/lustre_functions.sh
# FSx details
FSX_DNS="" # Needs to be adjusted.
MOUNT_NAME="" # Needs to be adjusted.
MOUNT_POINT="" # Needs to be adjusted.
# Function to check if Lustre is mounted
is_lustre_mounted() {
    mount | grep -q "type lustre"
}
# Function to mount Lustre
mount_lustre() {
    echo "Mounting Lustre filesystem..."
    mkdir -p ${MOUNT_POINT}
    mount -t lustre ${FSX_DNS}@tcp:${MOUNT_NAME} ${MOUNT_POINT}
    return $?
}
# Main execution
# Try to mount if not already mounted
if ! is_lustre_mounted; then
    echo "Lustre filesystem not mounted, attempting to mount..."
    mount_lustre
fi
# Wait for successful mount (up to 5 minutes)
for i in {1..30}; do
    if is_lustre_mounted; then
```

```

        echo "Lustre filesystem mounted, applying tunings..."
        apply_lustre_tunings
        exit 0
    fi
    echo "Waiting for Lustre filesystem to be mounted... (attempt $i/30)"
    sleep 10
done
echo "Timeout waiting for Lustre filesystem mount"
exit 1
EOF

# Create systemd service
cat << 'EOF' > /etc/systemd/system/lustre-tunings.service
[Unit]
Description=Apply Lustre Performance Tunings
After=network.target remote-fs.target
StartLimitIntervalSec=0
[Service]
Type=oneshot
ExecStart=/usr/local/bin/apply_lustre_tunings.sh
RemainAfterExit=yes
Restart=on-failure
RestartSec=30
[Install]
WantedBy=multi-user.target
EOF

    chmod +x /usr/local/bin/lustre_functions.sh
    chmod +x /usr/local/bin/apply_lustre_tunings.sh
    systemctl enable lustre-tunings.service
    systemctl start lustre-tunings.service
}
echo "Starting FSx for Lustre configuration..."
# 1. Install Lustre client
if grep -q 'VERSION="2"' /etc/os-release; then
    amazon-linux-extras install -y lustre
elif grep -q 'VERSION="2023"' /etc/os-release; then
    dnf install -y lustre-client
fi
check_success "Install Lustre client"
# 2. Apply network and RPC tunings
export PATH=$PATH:/usr/sbin
echo "Applying network and RPC tunings..."
if ! grep -q "options ptlrpc ptlrpcd_per_cpt_max" /etc/modprobe.d/modprobe.conf; then
    echo "options ptlrpc ptlrpcd_per_cpt_max=64" | tee -a /etc/modprobe.d/
modprobe.conf

```

```

    echo "options ksocklnd credits=2560" | tee -a /etc/modprobe.d/modprobe.conf
fi
# 3. Load Lustre modules
modprobe lustre
check_success "Load Lustre modules" || exit 1
# 4. Lustre Network Initialization
lctl network up
check_success "Initialize Lustre networking" || exit 1
# 5. Mount FSx filesystem
FSX_DNS="" # Needs to be adjusted.
MOUNT_NAME="" # Needs to be adjusted.
MOUNT_POINT="" # Needs to be adjusted.
if [ ! -z "$FSX_DNS" ] && [ ! -z "$MOUNT_NAME" ]; then
    mkdir -p $MOUNT_POINT
    mount -t lustre ${FSX_DNS}@tcp:/${MOUNT_NAME} ${MOUNT_POINT}
    check_success "Mount FSx filesystem"
fi
# 6. Apply tunings
apply_tunings
# 7. Verify tunings
verify_tunings
# 8. Setup persistence
setup_persistence
echo "FSx for Lustre configuration completed."
---MYBOUNDARY---

```

6. Saat membuat grup simpul Amazon EKS, pilih templat peluncuran ini. Untuk informasi selengkapnya, lihat [the section called “Buat”](#).

Informasi terkait

- [the section called “Menyebarkan driver”](#)
- [the section called “Optimalkan \(EFA\)”](#)
- [Amazon FSx untuk Kinerja Kilau](#)

Gunakan penyimpanan aplikasi berkinerja tinggi FSx untuk NetApp ONTAP

NetApp Trident menyediakan orkestrasi penyimpanan dinamis menggunakan driver yang sesuai dengan Container Storage Interface (CSI). Hal ini memungkinkan kluster Amazon EKS untuk

mengelola siklus hidup volume persisten (PVs) yang didukung oleh Amazon FSx untuk sistem file NetApp ONTAP. Perhatikan bahwa driver Amazon FSx untuk NetApp ONTAP CSI tidak kompatibel dengan Amazon EKS Hybrid Nodes. Untuk memulai, lihat [Menggunakan Trident dengan Amazon FSx untuk NetApp ONTAP di dokumentasi](#) NetApp Trident.

Amazon FSx untuk NetApp ONTAP adalah layanan penyimpanan yang memungkinkan Anda meluncurkan dan menjalankan sistem file ONTAP yang dikelola sepenuhnya di cloud. ONTAP NetApp adalah teknologi sistem file yang menyediakan serangkaian akses data dan kemampuan manajemen data yang diadopsi secara luas. FSx untuk ONTAP menyediakan fitur, kinerja, dan APIs sistem NetApp file lokal dengan kelincahan, skalabilitas, dan kesederhanaan layanan yang dikelola sepenuhnya. AWS Untuk informasi selengkapnya, lihat [Panduan Pengguna ONTAP FSx untuk](#).

Important

Jika Anda menggunakan Amazon FSx untuk NetApp ONTAP bersama driver Amazon EBS CSI untuk menyediakan volume EBS, Anda harus menentukan untuk tidak menggunakan perangkat EBS dalam file `multipath.conf` Untuk metode yang didukung, lihat [Daftar Hitam File Konfigurasi](#). Inilah contohnya.

```
defaults {
    user_friendly_names yes
    find_multipaths no
}
blacklist {
    device {
        vendor "NVME"
        product "Amazon Elastic Block Store"
    }
}
```

Gunakan penyimpanan data dengan Amazon FSx untuk OpenZFS

Amazon FSx untuk OpenZFS adalah layanan penyimpanan file yang dikelola sepenuhnya yang memudahkan pemindahan data AWS dari ZFS lokal atau server file berbasis Linux lainnya. Anda dapat melakukan ini tanpa mengubah kode aplikasi Anda atau bagaimana Anda mengelola data. Ini menawarkan penyimpanan file yang sangat andal, terukur, efisien, dan kaya fitur yang dibangun di atas sistem file OpenZFS open-source. Ini menggabungkan kemampuan ini dengan

kelincahan, skalabilitas, dan kesederhanaan layanan yang dikelola AWS sepenuhnya. Untuk informasi selengkapnya, lihat [Panduan Pengguna Amazon FSx for OpenZFS](#).

Driver FSx for OpenZFS Container Storage Interface (CSI) menyediakan antarmuka CSI yang memungkinkan cluster Amazon EKS mengelola siklus hidup volume OpenZFS. FSx Perhatikan bahwa driver Amazon FSx untuk OpenZFS CSI tidak kompatibel dengan Amazon EKS Hybrid Nodes. [Untuk menerapkan driver CSI FSx untuk OpenZFS ke cluster Amazon EKS Anda, lihat -driver aktif. aws-fsx-openzfs-csi](#) GitHub

Minimalkan latensi dengan Amazon File Cache

Amazon File Cache adalah cache berkecepatan tinggi AWS yang dikelola sepenuhnya yang digunakan untuk memproses data file, di mana pun data disimpan. Amazon File Cache secara otomatis memuat data ke dalam cache saat diakses untuk pertama kalinya dan merilis data saat tidak digunakan. Untuk informasi selengkapnya, lihat [Panduan Pengguna Cache File Amazon](#).

Driver Amazon File Cache Container Storage Interface (CSI) menyediakan antarmuka CSI yang memungkinkan cluster Amazon EKS mengelola siklus hidup cache file Amazon. Perhatikan bahwa driver CSI Cache File Amazon tidak kompatibel dengan Amazon EKS Hybrid Nodes. Untuk menerapkan driver CSI Cache File Amazon ke kluster Amazon EKS Anda, lihat [aws-file-cache-csi-driver](#) aktif. GitHub

Akses objek Amazon S3 dengan Mountpoint untuk driver Amazon S3 CSI

Dengan driver [Mountpoint for Amazon S3 Container Storage Interface \(CSI\)](#), aplikasi Kubernetes Anda dapat mengakses objek Amazon S3 melalui antarmuka sistem file, mencapai throughput agregat tinggi tanpa mengubah kode aplikasi apa pun. Dibangun di [Mountpoint untuk Amazon S3](#), driver CSI menghadirkan bucket Amazon S3 sebagai volume yang dapat diakses oleh kontainer di Amazon EKS dan cluster Kubernetes yang dikelola sendiri.

Pertimbangan

- Driver Mountpoint untuk Amazon S3 CSI saat ini tidak kompatibel dengan gambar kontainer berbasis Windows.
- Driver Mountpoint untuk Amazon S3 CSI saat ini tidak kompatibel dengan Amazon EKS Hybrid Nodes.

- Driver Mountpoint untuk Amazon S3 CSI tidak mendukung Fargate. AWS Namun, container yang berjalan di Amazon EC2 (baik dengan Amazon EKS atau instalasi Kubernetes kustom) didukung.
- Driver Mountpoint untuk Amazon S3 CSI hanya mendukung penyediaan statis. Penyediaan dinamis, atau pembuatan bucket baru, tidak didukung.

Note

Penyediaan statis mengacu pada penggunaan bucket Amazon S3 yang sudah ada yang ditentukan sebagai `bucketName` dalam `objekvolumeAttributes`. `PersistentVolume`
Untuk informasi selengkapnya, lihat [Penyediaan Statis aktif](#). GitHub

- Volume yang dipasang dengan driver Mountpoint untuk Amazon S3 CSI tidak mendukung semua fitur sistem file POSIX. Untuk detail tentang perilaku sistem file, lihat [Mountpoint untuk perilaku sistem file Amazon S3 aktif](#). GitHub

Untuk detail tentang penerapan driver, lihat [the section called “Menyebarkan driver”](#). Untuk detail tentang menghapus driver, lihat [the section called “Hapus driver”](#).

Terapkan Mountpoint untuk driver Amazon S3

Dengan driver [Mountpoint for Amazon S3 Container Storage Interface \(CSI\)](#), aplikasi Kubernetes Anda dapat mengakses objek Amazon S3 melalui antarmuka sistem file, mencapai throughput agregat tinggi tanpa mengubah kode aplikasi apa pun.

Prosedur ini akan menunjukkan kepada Anda cara menerapkan driver [Mountpoint untuk Amazon S3 CSI](#) Amazon EKS. Sebelum melanjutkan, silakan tinjau [Pertimbangan](#).

Prasyarat

- Penyedia OpenID Connect (OIDC) AWS Identity and Access Management (IAM) yang sudah ada untuk kluster Anda. Untuk menentukan apakah Anda sudah memiliki satu, atau harus membuat satu, lihat [the section called “Penyedia IAM OIDC”](#).
- Versi 2.12.3 atau yang lebih baru dari AWS CLI diinstal dan dikonfigurasi pada perangkat Anda atau AWS CloudShell
- Alat baris perintah `kubectl` diinstal pada perangkat Anda atau AWS CloudShell. Versinya bisa sama dengan atau hingga satu versi minor lebih awal atau lebih lambat dari versi Kubernetes dari kluster Anda. Misalnya, jika versi cluster Anda 1.29, Anda dapat menggunakan `kubectl`

versi 1.28, 1.29, atau 1.30 dengan itu. Untuk menginstal atau memutakhirkan kubectl, lihat [the section called “Siapkan kubectl dan eksctl”](#).

Langkah 1: Buat kebijakan IAM

Driver Mountpoint untuk Amazon S3 CSI memerlukan izin Amazon S3 untuk berinteraksi dengan sistem file Anda. Bagian ini menunjukkan cara membuat kebijakan IAM yang memberikan izin yang diperlukan.

Kebijakan contoh berikut mengikuti rekomendasi izin IAM untuk Mountpoint. Atau, Anda dapat menggunakan tautan kebijakan AWS terkelola: `iam/home?#/policies/arn:aws:iam::aws:policy/AmazonS3FullAccess` \$JSONEditor [AmazonS3, type="console "], tetapi kebijakan terkelola ini memberikan lebih banyak izin daripada yang diperlukan untuk MountpointFullAccess.

Untuk informasi selengkapnya tentang izin yang disarankan untuk Mountpoint, lihat [Izin IAM Mountpoint](#) pada [GitHub](#).

1. Buka konsol IAM di <https://console.aws.amazon.com/iam/>.
2. Di panel navigasi di sebelah kiri, pilih Kebijakan.
3. Pada halaman Kebijakan, pilih Buat kebijakan.
4. Untuk editor Kebijakan, pilih JSON.
5. Di bawah Editor kebijakan, salin dan tempel berikut ini:

Important

Ganti `amzn-s3-demo-bucket1` dengan nama bucket Amazon S3 Anda sendiri.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "MountpointFullBucketAccess",
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket"
      ],
      "Resource": [
```

```

        "arn:aws: s3::amzn-s3-demo-bucket1"
    ],
    },
    {
        "Sid": "MountpointFullObjectAccess",
        "Effect": "Allow",
        "Action": [
            "s3:GetObject",
            "s3:PutObject",
            "s3:AbortMultipartUpload",
            "s3:DeleteObject"
        ],
        "Resource": [
            "arn:aws: s3::amzn-s3-demo-bucket1/*"
        ]
    }
]
}

```

Bucket direktori, diperkenalkan dengan kelas penyimpanan Amazon S3 Express One Zone, menggunakan mekanisme otentikasi yang berbeda dari bucket tujuan umum. Alih-alih menggunakan `s3:*` tindakan, Anda harus menggunakan `s3express:CreateSession` tindakan. Untuk informasi tentang bucket direktori, lihat [Bucket direktori di Panduan](#) Pengguna Amazon S3.

Di bawah ini adalah contoh kebijakan hak istimewa paling tidak yang akan Anda gunakan untuk bucket direktori.

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3express:CreateSession",
            "Resource": "arn:aws: s3express:us-west-2:111122223333:bucket/amzn-s3-
demo-bucket1--usw2-az1--x-s3"
        }
    ]
}

```

6. Pilih Berikutnya.

7. Pada halaman Tinjau dan buat, beri nama kebijakan Anda. Contoh panduan ini menggunakan nama. `AmazonS3CSIDriverPolicy`

8. Pilih Buat kebijakan.

Langkah 2: Buat peran IAM

Driver Mountpoint untuk Amazon S3 CSI memerlukan izin Amazon S3 untuk berinteraksi dengan sistem file Anda. Bagian ini menunjukkan cara membuat peran IAM untuk mendelegasikan izin ini. Untuk membuat peran ini, Anda dapat menggunakan salah satu alat ini:

- [the section called “eksctl”](#)
- [the section called “AWS Management Console”](#)
- [the section called “AWS CLI”](#)

Note

Kebijakan IAM AmazonS3CSIDriverPolicy dibuat di bagian sebelumnya.

eksctl

Untuk membuat peran IAM driver Mountpoint untuk Amazon S3 CSI Anda dengan **eksctl**

Untuk membuat peran IAM dan akun layanan Kubernetes, jalankan perintah berikut. Perintah ini juga melampirkan kebijakan AmazonS3CSIDriverPolicy IAM ke peran, membuat anotasi akun layanan Kubernetes (s3-csi-controller-sa) dengan Nama Sumber Daya Amazon (ARN) peran IAM, dan menambahkan nama akun layanan Kubernetes ke kebijakan kepercayaan untuk peran IAM.

```
CLUSTER_NAME=my-cluster
REGION=region-code
ROLE_NAME=AmazonEKS_S3_CSI_DriverRole
POLICY_ARN=AmazonEKS_S3_CSI_DriverRole_ARN
eksctl create iamserviceaccount \
  --name s3-csi-driver-sa \
  --namespace kube-system \
  --cluster $CLUSTER_NAME \
  --attach-policy-arn $POLICY_ARN \
  --approve \
  --role-name $ROLE_NAME \
  --region $REGION \
  --role-only
```

AWS Management Console

1. Buka konsol IAM di <https://console.aws.amazon.com/iam/>.
2. Di panel navigasi sebelah kiri, pilih Peran.
3. Pada halaman Peran, pilih Buat peran.
4. Pada halaman Pilih entitas tepercaya, lakukan hal berikut:
 - a. Di bagian Jenis entitas tepercaya, pilih Identitas web.
 - b. Untuk penyedia Identity, pilih URL penyedia OpenID Connect untuk kluster Anda (seperti yang ditunjukkan di bawah Ikhtisar di Amazon EKS).

Jika tidak URLs ditampilkan, tinjau [Prasyarat](#).

- c. Untuk Audiens, pilih `sts.amazonaws.com`.
 - d. Pilih Berikutnya.
5. Pada halaman Tambahkan izin, lakukan hal berikut:
 - a. Di kotak Filter kebijakan, masukkan Kebijakan AmazonS3CSIDriver.



Note

Kebijakan ini dibuat di bagian sebelumnya.

- b. Pilih kotak centang di sebelah kiri AmazonS3CSIDriverPolicy hasil yang dikembalikan dalam pencarian.
 - c. Pilih Berikutnya.
6. Pada halaman Nama, tinjau, dan buat, lakukan hal berikut:
 - a. Untuk nama Peran, masukkan nama unik untuk peran Anda, seperti `DriverRole Amazoneks_S3_CSI_`.
 - b. Di bawah Tambahkan tag (Opsional), tambahkan metadata ke peran dengan melampirkan tag sebagai pasangan nilai kunci. Untuk informasi selengkapnya tentang penggunaan tanda di IAM, lihat [Menandai sumber daya IAM](#) di Panduan Pengguna IAM.
 - c. Pilih Buat peran.
 7. Setelah peran dibuat, pilih peran di konsol untuk dibuka, dan kemudian diedit.
 8. Pilih tab Trust relationship, lalu pilih Edit trust policy.
 9. Temukan baris yang serupa dengan yang berikut ini:

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud":
"sts.amazonaws.com"
```

Tambahkan koma ke akhir baris sebelumnya, lalu tambahkan baris berikut setelahnya.

Ganti *region-code* dengan AWS Wilayah tempat cluster Anda berada. Ganti *EXAMPLED539D4633E53DE1B71EXAMPLE* dengan ID penyedia OIDC cluster Anda.

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub":
"system:serviceaccount:kube-system:s3-csi-driver-sa"
```

10Pastikan Condition operator disetel ke "StringEquals".

11Pilih Perbarui kebijakan untuk menyelesaikan.

AWS CLI

1. Lihat URL penyedia OIDC untuk klaster Anda. Ganti *my-cluster* dengan nama klaster Anda. Jika output dari perintah adalah None, tinjau [Prasyarat](#).

```
aws eks describe-cluster --name my-cluster --query "cluster.identity.oidc.issuer" --
output text
```

Contoh output adalah sebagai berikut.

```
https://oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE
```

2. Buat IAM role, yang memberikan tindakan AssumeRoleWithWebIdentity untuk akun layanan Kubernetes.
 - a. Salin konten berikut ke file bernama `aws-s3-csi-driver-trust-policy.json`. Ganti *111122223333* dengan ID akun Anda. Ganti *EXAMPLED539D4633E53DE1B71EXAMPLE* dan *region-code* dengan nilai yang dikembalikan pada langkah sebelumnya.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
```

```

    "Federated": "arn:aws: iam::111122223333:oidc-provider/oidc.eks.region-
code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
  },
  "Action": "sts:AssumeRoleWithWebIdentity",
  "Condition": {
    "StringEquals": {
      "oidc.eks.region-code.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-system:s3-csi-
driver-sa",
      "oidc.eks.region-code.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com"
    }
  }
}
]
}

```

- b. Buat peran. Anda dapat mengubah *AmazonEKS_S3_CSI_DriverRole* ke nama yang berbeda, tetapi jika Anda melakukannya, pastikan untuk mengubahnya juga dalam langkah berikutnya.

```

aws iam create-role \
  --role-name AmazonEKS_S3_CSI_DriverRole \
  --assume-role-policy-document file://"aws-s3-csi-driver-trust-policy.json"

```

3. Lampirkan kebijakan IAM yang dibuat sebelumnya ke peran dengan perintah berikut.

```

aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonS3CSIDriverPolicy \
  --role-name AmazonEKS_S3_CSI_DriverRole

```

Note

Kebijakan IAM AmazonS3CSIDriverPolicy dibuat di bagian sebelumnya.

4. Lewati langkah ini jika Anda menginstal driver sebagai add-on Amazon EKS. Untuk instalasi driver yang dikelola sendiri, buat akun layanan Kubernetes yang dianotasi dengan ARN dari peran IAM yang Anda buat.

- a. Simpan konten berikut ini ke file bernama `mountpoint-s3-service-account.yaml`. Ganti *111122223333* dengan ID akun Anda.

```
---
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    app.kubernetes.io/name: aws-mountpoint-s3-csi-driver
  name: mountpoint-s3-csi-controller-sa
  namespace: kube-system
  annotations:
    eks.amazonaws.com/role-arn: arn:aws:iam::111122223333:role/
    AmazonEKS_S3_CSI_DriverRole
```

- b. Buat akun layanan Kubernetes di klaster Anda. Akun layanan Kubernetes (mountpoint-s3-csi-controller-sa) dianotasi dengan peran IAM yang Anda buat bernama.

AmazonEKS_S3_CSI_DriverRole

```
kubectl apply -f mountpoint-s3-service-account.yaml
```

 Note

Ketika Anda menyebarkan plugin dalam prosedur ini, itu membuat dan dikonfigurasi untuk menggunakan akun layanan bernama `s3-csi-driver-sa`.

Langkah 3: Instal Mountpoint untuk driver Amazon S3 CSI

Anda dapat menginstal driver Mountpoint untuk Amazon S3 CSI melalui add-on Amazon EKS. Anda dapat menggunakan alat berikut untuk menambahkan add-on ke cluster Anda:

- [the section called “eksctl”](#)
- [the section called “AWS Management Console”](#)
- [the section called “AWS CLI”](#)

Atau, Anda dapat menginstal driver Mountpoint untuk Amazon S3 CSI sebagai instalasi yang dikelola sendiri. Untuk petunjuk cara melakukan instalasi yang dikelola sendiri, lihat [Instalasi aktif](#). GitHub

Mulai dari `v1.8.0`, Anda dapat mengonfigurasi taint untuk mentolerir Pod driver CSI. Untuk melakukan ini, tentukan satu set taint khusus untuk ditoleransi `node.tolerations` atau tolerasikan

semua noda. `node.tolerateAllTaints` Untuk informasi selengkapnya, lihat [Taints and Tolerations dalam dokumentasi](#) Kubernetes.

`eksctl`

Untuk menambahkan add-on Amazon S3 CSI menggunakan `eksctl`

Jalankan perintah berikut. Ganti *my-cluster* dengan nama kluster Anda, *111122223333* dengan ID akun Anda, dan *AmazonEKS_S3_CSI_DriverRole* dengan nama [peran IAM yang dibuat sebelumnya](#).

```
eksctl create addon --name aws-mountpoint-s3-csi-driver --cluster my-cluster \
  --service-account-role-arn arn:aws:iam::111122223333:role/
AmazonEKS_S3_CSI_DriverRole --force
```

Jika Anda menghapus *--force* opsi dan salah satu pengaturan add-on Amazon EKS bertentangan dengan pengaturan yang ada, pembaruan add-on Amazon EKS gagal, dan Anda menerima pesan kesalahan untuk membantu Anda menyelesaikan konflik. Sebelum menentukan opsi ini, pastikan add-on Amazon EKS tidak mengelola pengaturan yang perlu Anda kelola, karena pengaturan tersebut ditimpa dengan opsi ini. Untuk informasi selengkapnya tentang opsi lain untuk setelan ini, lihat [Addons](#) dalam `eksctl` dokumentasi. Untuk informasi selengkapnya tentang pengelolaan lapangan Amazon EKS Kubernetes, lihat [the section called “Bidang yang dapat Anda sesuaikan”](#)

Anda dapat menyesuaikan `eksctl` melalui file konfigurasi. Untuk informasi selengkapnya, lihat [Bekerja dengan nilai konfigurasi](#) dalam `eksctl` dokumentasi. Contoh berikut menunjukkan bagaimana mentolerir semua noda.

```
# config.yaml
...

addons:
- name: aws-mountpoint-s3-csi-driver
  serviceAccountRoleARN: arn:aws:iam::111122223333:role/AmazonEKS_S3_CSI_DriverRole
  configurationValues: |-
    node:
      tolerateAllTaints: true
```

AWS Management Console

1. Buka [konsol Amazon EKS](#).

2. Pada panel navigasi sebelah kiri, pilih Klaster.
3. Pilih nama cluster yang ingin Anda konfigurasikan add-on Mountpoint untuk Amazon S3 CSI.
4. Pilih tab Add-ons.
5. Pilih Get more add-ons
6. Pada halaman Pilih add-on, lakukan hal berikut:
 - a. Di bagian Amazon EKS-Addons, pilih kotak centang Mountpoint for Amazon S3 CSI Driver.
 - b. Pilih Berikutnya.
7. Pada halaman Configure selected add-ons settings, lakukan hal berikut:
 - a. Pilih Versi yang ingin Anda gunakan.
 - b. Untuk peran Pilih IAM, pilih nama peran IAM yang Anda lampirkan ke kebijakan IAM driver Mountpoint for Amazon S3 CSI.
 - c. (Opsional) Perbarui metode resolusi Konflik setelah memperluas pengaturan konfigurasi Opsional. Jika Anda memilih Override, satu atau beberapa pengaturan untuk add-on yang ada dapat ditimpa dengan pengaturan add-on Amazon EKS. Jika Anda tidak mengaktifkan opsi ini dan ada konflik dengan pengaturan yang ada, operasi gagal. Anda dapat menggunakan pesan kesalahan yang dihasilkan untuk memecahkan masalah konflik. Sebelum memilih opsi ini, pastikan add-on Amazon EKS tidak mengelola pengaturan yang perlu Anda kelola sendiri.
 - d. (Opsional) Konfigurasikan toleransi di bidang Nilai konfigurasi setelah memperluas pengaturan konfigurasi opsional.
 - e. Pilih Berikutnya.
8. Pada halaman Review dan add, pilih Create. Setelah penginstalan add-on selesai, Anda melihat add-on yang diinstal.

AWS CLI

Untuk menambahkan add-on Mountpoint untuk Amazon S3 CSI menggunakan CLI AWS

Jalankan perintah berikut. Ganti *my-cluster* dengan nama klaster Anda, *111122223333* dengan ID akun Anda, dan *AmazonEKS_S3_CSI_DriverRole* dengan nama peran yang dibuat sebelumnya.

```
aws eks create-addon --cluster-name my-cluster --addon-name aws-mountpoint-s3-csi-driver \
  --service-account-role-arn arn:aws:iam::111122223333:role/AmazonEKS_S3_CSI_DriverRole
```

Anda dapat menyesuaikan perintah dengan `--configuration-values` bendera. Contoh alternatif berikut menunjukkan bagaimana mentolerir semua noda.

```
aws eks create-addon --cluster-name my-cluster --addon-name aws-mountpoint-s3-csi-  
driver \  
  --service-account-role-arn arn:aws:iam::111122223333:role/  
AmazonEKS_S3_CSI_DriverRole \  
  --configuration-values '{"node":{"tolerateAllTaints":true}}'
```

Langkah 4: Konfigurasi Mountpoint untuk Amazon S3

Dalam kebanyakan kasus, Anda dapat mengonfigurasi Mountpoint untuk Amazon S3 hanya dengan nama bucket. Untuk petunjuk tentang mengonfigurasi Mountpoint untuk Amazon S3, lihat Mengonfigurasi [Mountpoint untuk Amazon S3 aktif](#). GitHub

Langkah 5: Menyebarkan aplikasi sampel

Anda dapat menerapkan penyediaan statis ke driver di bucket Amazon S3 yang ada. Untuk informasi selengkapnya, lihat [Penyediaan statis aktif](#). GitHub

Hapus add-on Mountpoint untuk Amazon S3 Amazon EKS

Anda memiliki dua opsi untuk menghapus driver [Mountpoint untuk Amazon S3](#) CSI.

- Pertahankan perangkat lunak add-on di cluster Anda — Opsi ini menghapus pengelolaan Amazon EKS dari pengaturan apa pun. Ini juga menghapus kemampuan Amazon EKS untuk memberi tahu Anda tentang pembaruan dan secara otomatis memperbarui add-on Amazon EKS setelah Anda memulai pembaruan. Namun, ini mempertahankan perangkat lunak add-on di cluster Anda. Opsi ini menjadikan add-on instalasi yang dikelola sendiri, bukan add-on Amazon EKS. Dengan opsi ini, tidak ada downtime untuk add-on. Perintah dalam prosedur ini menggunakan opsi ini.
- Hapus perangkat lunak add-on sepenuhnya dari klaster Anda — Kami menyarankan Anda menghapus add-on Amazon EKS dari klaster Anda hanya jika tidak ada sumber daya di klaster Anda yang bergantung padanya. Untuk melakukan opsi ini, hapus `--preserve` dari perintah yang Anda gunakan dalam prosedur ini.

Jika add-on memiliki akun IAM yang terkait dengannya, akun IAM tidak akan dihapus.

Anda dapat menggunakan alat berikut untuk menghapus add-on Amazon S3 CSI:

- [the section called “eksctl”](#)
- [the section called “AWS Management Console”](#)
- [the section called “AWS CLI”](#)

eksctl

Untuk menghapus add-on Amazon S3 CSI menggunakan **eksctl**

Ganti *my-cluster* dengan nama cluster Anda, lalu jalankan perintah berikut.

```
eksctl delete addon --cluster my-cluster --name aws-mountpoint-s3-csi-driver --preserve
```

AWS Management Console

1. Buka [konsol Amazon EKS](#).
2. Pada panel navigasi sebelah kiri, pilih Klaster.
3. Pilih nama cluster yang ingin Anda hapus add-on Amazon EBS CSI.
4. Pilih tab Add-ons.
5. Pilih Mountpoint untuk Driver Amazon S3 CSI.
6. Pilih Hapus.
7. Dalam Hapus: Konfirmasi aws-mountpoint-s 3-csi-driver kotak dialog, lakukan hal berikut:
 - a. Jika Anda ingin Amazon EKS berhenti mengelola pengaturan untuk add-on, pilih Pertahankan di klaster. Lakukan ini jika Anda ingin mempertahankan perangkat lunak add-on di cluster Anda. Ini agar Anda dapat mengelola semua pengaturan add-on sendiri.
 - b. Masukkan `aws-mountpoint-s3-csi-driver`.
 - c. Pilih Hapus.

AWS CLI

Untuk menghapus add-on Amazon S3 CSI menggunakan CLI AWS

Ganti *my-cluster* dengan nama cluster Anda, lalu jalankan perintah berikut.

```
aws eks delete-addon --cluster-name my-cluster --addon-name aws-mountpoint-s3-csi-driver --preserve
```

Aktifkan fungsionalitas snapshot untuk volume CSI

Fungsionalitas snapshot memungkinkan point-in-time salinan data Anda. Agar kemampuan ini dapat bekerja di Kubernetes, Anda memerlukan driver CSI dengan dukungan snapshot (seperti driver Amazon EBS CSI) dan pengontrol snapshot CSI. Pengontrol snapshot tersedia baik sebagai add-on terkelola Amazon EKS atau sebagai instalasi yang dikelola sendiri.

Berikut adalah beberapa hal yang perlu dipertimbangkan saat menggunakan pengontrol snapshot CSI.

- Pengontrol snapshot harus diinstal bersama driver CSI dengan fungsionalitas snapshot. Untuk petunjuk penginstalan driver Amazon EBS CSI, lihat [the section called “Amazon EBS”](#)
- Kubernetes tidak mendukung snapshot volume yang disajikan melalui migrasi CSI, seperti volume Amazon EBS yang menggunakan penyedia with. `StorageClass` `kubernetes.io/aws-ebs` Volume harus dibuat dengan `StorageClass` yang mereferensikan penyedia driver CSI, `ebs.csi.aws.com`
- Amazon EKS Auto Mode tidak termasuk pengontrol snapshot. Kemampuan penyimpanan Mode Otomatis EKS kompatibel dengan pengontrol snapshot.

Kami menyarankan Anda menginstal pengontrol snapshot CSI melalui add-on terkelola Amazon EKS. Add-on ini mencakup definisi sumber daya kustom (CRDs) yang diperlukan untuk membuat dan mengelola snapshot di Amazon EKS. Untuk menambahkan add-on Amazon EKS ke cluster Anda, lihat [the section called “Buat add-on”](#). Untuk informasi selengkapnya tentang add-on, lihat [the section called “Add-on Amazon EKS”](#).

Atau, jika Anda menginginkan instalasi yang dikelola sendiri dari pengontrol snapshot CSI, lihat [Penggunaan](#) di Kubernetes upstream aktif. `external-snapshotter` GitHub

Konfigurasi jaringan untuk kluster Amazon EKS

Cluster Amazon EKS Anda dibuat dalam VPC. Jaringan pod disediakan oleh plugin Amazon VPC Container Network Interface (CNI) untuk node yang berjalan pada infrastruktur. AWS Jika Anda menjalankan node pada infrastruktur Anda sendiri, lihat [the section called “Konfigurasi CNI”](#). Bab ini mencakup topik-topik berikut untuk mempelajari lebih lanjut tentang jaringan untuk cluster Anda.

Topik

- [Tambahkan Subnet VPC yang ada ke cluster Amazon EKS dari konsol manajemen](#)
- [Lihat persyaratan jaringan Amazon EKS untuk VPC dan subnet](#)
- [Buat VPC Amazon untuk cluster Amazon EKS Anda](#)
- [Lihat persyaratan grup keamanan Amazon EKS untuk cluster](#)
- [Kelola add-on jaringan untuk cluster Amazon EKS](#)

Tambahkan Subnet VPC yang ada ke cluster Amazon EKS dari konsol manajemen

1. Arahkan ke klaster Anda di konsol manajemen
2. Dari tab Jaringan pilih Kelola Sumber Daya VPC
3. Dari dropdown Subnet, pilih subnet tambahan dari VPC cluster Anda.

Untuk membuat Subnet VPC baru:

- [Tinjau Persyaratan Subnet EKS](#)
- Lihat [Membuat Subnet](#) di Panduan Pengguna Amazon Virtual Private Cloud.

Lihat persyaratan jaringan Amazon EKS untuk VPC dan subnet

Saat membuat cluster, Anda menentukan [VPC](#) dan setidaknya dua subnet yang berada di Availability Zone yang berbeda. Topik ini memberikan gambaran umum tentang persyaratan dan pertimbangan khusus Amazon EKS untuk VPC dan subnet yang Anda gunakan dengan klaster Anda. Jika Anda tidak memiliki VPC untuk digunakan dengan Amazon EKS, lihat [the section called “Buat VPC”](#). Jika Anda membuat cluster lokal atau diperluas di AWS Outposts, lihat [the section called “Buat VPC dan](#)

[subnet](#)” alih-alih topik ini. Konten dalam topik ini berlaku untuk cluster Amazon EKS dengan node hibrida. Untuk persyaratan jaringan tambahan untuk node hybrid, lihat [the section called “Siapkan jaringan”](#).

Persyaratan dan pertimbangan VPC

Saat Anda membuat klaster, VPC yang Anda tentukan harus memenuhi persyaratan dan pertimbangan berikut:

- VPC harus memiliki cukup banyak alamat IP yang tersedia untuk cluster, node apa pun, dan sumber daya Kubernetes lainnya yang ingin Anda buat. Jika VPC yang ingin Anda gunakan tidak memiliki jumlah alamat IP yang cukup, cobalah untuk menambah jumlah alamat IP yang tersedia.

Anda dapat melakukan ini dengan memperbarui konfigurasi cluster untuk mengubah subnet dan grup keamanan yang digunakan cluster. Anda dapat memperbarui dari AWS Management Console, versi terbaru dari AWS CLI, AWS CloudFormation, dan `eksctl` versi `v0.164.0-rc.0` atau yang lebih baru. Anda mungkin perlu melakukan ini untuk menyediakan subnet dengan lebih banyak alamat IP yang tersedia untuk berhasil meningkatkan versi cluster.

Important

Semua subnet yang Anda tambahkan harus dalam set yang sama AZs seperti yang awalnya disediakan saat Anda membuat cluster. Subnet baru harus memenuhi semua persyaratan lainnya, misalnya mereka harus memiliki alamat IP yang memadai. Misalnya, asumsikan bahwa Anda membuat cluster dan menentukan empat subnet. Dalam urutan yang Anda tentukan, subnet pertama berada di `us-west-2a` Availability Zone, subnet kedua dan ketiga berada di `us-west-2b` Availability Zone, dan subnet keempat berada di `us-west-2c` Availability Zone. Jika Anda ingin mengubah subnet, Anda harus menyediakan setidaknya satu subnet di masing-masing dari tiga Availability Zone, dan subnet harus dalam VPC yang sama dengan subnet asli.

Jika Anda membutuhkan lebih banyak alamat IP daripada blok CIDR di VPC, Anda dapat menambahkan blok CIDR tambahan [dengan mengaitkan blok Classless Inter-Domain Routing \(CIDR\) tambahan dengan](#) VPC Anda. Anda dapat mengaitkan blok CIDR pribadi (RFC 1918) dan publik (non-RFC 1918) ke VPC Anda baik sebelum atau setelah Anda membuat cluster Anda.

Anda dapat menggunakan blok CIDR baru segera setelah Anda menambahkannya. Namun, karena bidang kontrol mengenali blok CIDR baru hanya setelah rekonsiliasi selesai, diperlukan cluster hingga lima jam agar blok CIDR yang Anda kaitkan dengan VPC dikenali. Kemudian Anda dapat menjalankan `kubectl logs` perintah `kubectl exec` dan. Selain itu, jika Anda memiliki Pod yang beroperasi sebagai backend webhook, maka Anda harus menunggu rekonsiliasi bidang kontrol selesai.

Anda dapat menghemat penggunaan alamat IP dengan menggunakan gateway transit dengan VPC layanan bersama. Untuk informasi selengkapnya, lihat [Terisolasi VPCs dengan layanan bersama](#) dan [pola konservasi alamat IP yang dapat dirutekan Amazon EKS VPC dalam jaringan hibrida](#).

- Jika Anda ingin Kubernetes menetapkan IPv6 alamat ke Pod dan layanan, kaitkan IPv6 blok CIDR dengan VPC Anda. Untuk informasi selengkapnya, lihat [Mengaitkan blok IPv6 CIDR dengan VPC Anda](#) di Panduan Pengguna Amazon VPC. Anda tidak dapat menggunakan IPv6 alamat dengan Pod dan layanan yang berjalan pada node hybrid dan Anda tidak dapat menggunakan node hybrid dengan cluster yang dikonfigurasi dengan keluarga alamat IPv6 IP.
- VPC harus memiliki DNS nama host dan DNS dukungan resolusi. Jika tidak, node tidak dapat mendaftar ke cluster Anda. Untuk informasi selengkapnya, lihat [atribut DNS untuk VPC Anda](#) di Panduan Pengguna Amazon VPC.
- VPC mungkin memerlukan titik akhir VPC menggunakan AWS PrivateLink Untuk informasi selengkapnya, lihat [the section called “Persyaratan dan pertimbangan subnet”](#).

Jika Anda membuat klaster dengan Kubernetes 1.14 atau sebelumnya, Amazon EKS menambahkan tag berikut ke VPC Anda:

Kunci	Nilai
<code>kubernetes.io/cluster/</code> <i>my-cluster</i>	owned

Tag ini hanya digunakan oleh Amazon EKS. Anda dapat menghapus tag tanpa memengaruhi layanan Anda. Ini tidak digunakan dengan cluster yang versi 1.15 atau lebih baru.

Persyaratan dan pertimbangan subnet

Saat Anda membuat klaster, Amazon EKS membuat 2-4 [antarmuka jaringan elastis](#) di subnet yang Anda tentukan. Antarmuka jaringan ini memungkinkan komunikasi antara cluster Anda dan VPC Anda. Antarmuka jaringan ini juga mengaktifkan fitur Kubernetes seperti `dan. kubectl exec kubectl logs` Setiap antarmuka jaringan Amazon EKS yang dibuat memiliki teks Amazon EKS *cluster-name* dalam deskripsinya.

Amazon EKS dapat membuat antarmuka jaringannya di subnet apa pun yang Anda tentukan saat membuat cluster. Anda dapat mengubah subnet Amazon EKS yang membuat antarmuka jaringannya setelah cluster Anda dibuat. Saat Anda memperbarui klaster versi Kubernetes, Amazon EKS menghapus antarmuka jaringan asli yang dibuatnya, dan membuat antarmuka jaringan baru. Antarmuka jaringan ini dapat dibuat dalam subnet yang sama dengan antarmuka jaringan asli atau dalam subnet yang berbeda dari antarmuka jaringan asli. Untuk mengontrol antarmuka jaringan subnet mana yang dibuat, Anda dapat membatasi jumlah subnet yang Anda tentukan hanya dua saat Anda membuat cluster atau memperbarui subnet setelah membuat cluster.

Persyaratan subnet untuk cluster

[Subnet](#) yang Anda tentukan saat membuat atau memperbarui klaster harus memenuhi persyaratan berikut:

- Subnet masing-masing harus memiliki setidaknya enam alamat IP untuk digunakan oleh Amazon EKS. Namun, kami merekomendasikan setidaknya 16 alamat IP.
- Subnet harus berada di setidaknya dua Availability Zone yang berbeda.
- Subnet tidak dapat berada di AWS Outposts atau Wavelength. AWS Namun, jika Anda memilikinya di VPC Anda, Anda dapat menerapkan node yang dikelola sendiri dan sumber daya Kubernetes ke jenis subnet ini. Untuk informasi selengkapnya tentang node yang dikelola sendiri, lihat [the section called “Simpul yang dikelola sendiri”](#).
- Subnet dapat bersifat publik atau pribadi. Namun, kami menyarankan Anda menentukan subnet pribadi, jika memungkinkan. Subnet publik adalah subnet dengan tabel rute yang mencakup rute ke [gateway internet](#), sedangkan subnet pribadi adalah subnet dengan tabel rute yang tidak menyertakan rute ke gateway internet.
- Subnet tidak dapat berada di Availability Zone berikut:

AWS Wilayah	Nama wilayah	Zona Ketersediaan Dilarang IDs
us-east-1	AS Timur (Virginia Utara)	use1-az3
us-west-1	AS Barat (California Utara)	usw1-az2
ca-central-1	Kanada (Pusat)	cac1-az3

Penggunaan keluarga alamat IP berdasarkan komponen

Tabel berikut berisi keluarga alamat IP yang digunakan oleh setiap komponen Amazon EKS. Anda dapat menggunakan terjemahan alamat jaringan (NAT) atau sistem kompatibilitas lainnya untuk terhubung ke komponen ini dari alamat IP sumber dalam keluarga dengan nilai “Tidak” untuk entri tabel.

Fungsionalitas dapat berbeda tergantung pada pengaturan keluarga IP (ipFamily) cluster. Pengaturan ini mengubah jenis alamat IP yang digunakan untuk blok CIDR yang Kubernetes tetapkan ke Services. Sebuah cluster dengan nilai pengaturan IPv4 disebut sebagai IPv4 cluster, dan cluster dengan nilai pengaturan IPv6 disebut sebagai IPv6 cluster.

Komponen	IPv4 alamat	IPv6 alamat	Alamat tumpukan ganda
Titik akhir publik EKS API	Ya ^{1,3}	Ya ^{1,3}	Ya ^{1,3}
EKS API VPC titik akhir	Ya	Tidak	Tidak
Titik akhir publik EKS Auth API (Identitas Pod EKS)	Ya ¹	Ya ¹	Ya ¹
Titik akhir VPC EKS Auth API (Identitas Pod EKS)	Ya ¹	Ya ¹	Ya ¹

Komponen	IPv4 alamat	IPv6 alamat	Alamat tumpukan ganda
IPv4 ¹ Titik akhir publik klaster Kubernetes 2	Ya	Tidak	Tidak
IPv4 ¹ Titik akhir pribadi klaster Kubernetes 2	Ya	Tidak	Tidak
IPv6 ¹ Titik akhir publik klaster Kubernetes 2	Ya ^{1,4}	Ya ^{1,4}	Ya ⁴
IPv6 ¹ Titik akhir pribadi klaster Kubernetes 2	Ya ^{1,4}	Ya ^{1,4}	Ya ⁴
Subnet klaster Kubernetes	Ya ²	Tidak	Ya ²
Node Alamat IP Primer	Ya ²	Tidak	Ya ²
Kisaran Cluster CIDR untuk alamat IP Layanan	Ya ²	Ya ²	Tidak
Alamat IP Pod dari VPC CNI	Ya ²	Ya ²	Tidak
Penerbit IRSA OIDC URLs	Ya ^{1,3}	Ya ^{1,3}	Ya ^{1,3}

Note

¹ Titik akhir adalah tumpukan ganda dengan keduanya IPv4 dan IPv6 alamat. Aplikasi Anda di luar AWS, node Anda untuk cluster, dan pod Anda di dalam cluster dapat mencapai titik akhir ini dengan salah satu IPv4 atau IPv6.

² Anda memilih antara IPv4 cluster dan IPv6 cluster dalam pengaturan IP family (ipFamily) cluster saat Anda membuat cluster dan ini tidak dapat diubah. Sebagai gantinya,

Anda harus memilih pengaturan yang berbeda saat membuat kluster lain dan memigrasikan beban kerja Anda.

³ Titik akhir tumpukan ganda diperkenalkan pada Agustus 2024. Untuk menggunakan titik akhir dual-stack dengan AWS CLI, lihat konfigurasi titik akhir [Dual-stack dan FIPS di Panduan Referensi Alat dan](#). AWS SDKs Berikut ini mencantumkan endpoint baru:

Titik akhir publik EKS API

```
eks.region.api.aws
```

Penerbit IRSA OIDC URLs

```
oidc-eks.region.api.aws
```

⁴ Titik akhir cluster dual-stack diperkenalkan pada Oktober 2024. EKS membuat titik akhir berikut untuk cluster baru yang dibuat setelah tanggal ini dan yang memilih IPv6 dalam pengaturan keluarga IP (IPFamily) cluster:

EKS cluster titik akhir publik/pribadi

```
eks-cluster.region.api.aws
```

Persyaratan subnet untuk node

Anda dapat menerapkan node dan sumber daya Kubernetes ke subnet yang sama dengan yang Anda tentukan saat membuat kluster Anda. Namun, ini tidak perlu. Ini karena Anda juga dapat menerapkan node dan sumber daya Kubernetes ke subnet yang tidak Anda tentukan saat Anda membuat kluster. Jika Anda menerapkan node ke subnet yang berbeda, Amazon EKS tidak membuat antarmuka jaringan cluster di subnet tersebut. Setiap subnet yang Anda gunakan untuk menggunakan node dan sumber daya Kubernetes harus memenuhi persyaratan berikut:

- Subnet harus memiliki cukup alamat IP yang tersedia untuk menyebarkan semua node dan sumber daya Kubernetes Anda.
- Jika Anda ingin Kubernetes menetapkan IPv6 alamat ke Pod dan layanan, maka Anda harus memiliki satu blok CIDR dan satu blok IPv6 CIDR yang terkait dengan IPv4 subnet Anda. Untuk informasi selengkapnya, lihat [Mengaitkan blok IPv6 CIDR dengan subnet Anda](#) di Panduan Pengguna Amazon VPC. Tabel rute yang terkait dengan subnet harus menyertakan rute ke IPv4 dan IPv6 alamat. Untuk informasi selengkapnya, lihat [Rute](#) di Panduan Pengguna Amazon VPC.

Pod hanya diberi IPv6 alamat. Namun antarmuka jaringan yang dibuat Amazon EKS untuk kluster Anda dan node Anda diberi alamat IPv4 dan IPv6 alamat.

- Jika Anda memerlukan akses masuk dari internet ke Pod Anda, pastikan untuk memiliki setidaknya satu subnet publik dengan alamat IP yang cukup tersedia untuk menyebarkan load balancer dan ingress. Anda dapat menyebarkan penyeimbang beban ke subnet publik. Load balancer dapat memuat keseimbangan ke Pod di subnet pribadi atau publik. Sebaiknya gunakan node Anda ke subnet pribadi, jika memungkinkan.
- Jika Anda berencana untuk menyebarkan node ke subnet publik, subnet harus menetapkan alamat atau alamat publik secara otomatis IPv4. IPv6 Jika Anda menyebarkan node ke subnet pribadi yang memiliki blok IPv6 CIDR terkait, subnet pribadi juga harus menetapkan alamat secara otomatis. IPv6 Jika Anda menggunakan AWS CloudFormation template yang disediakan oleh Amazon EKS untuk menerapkan VPC Anda setelah 26 Maret 2020, pengaturan ini diaktifkan. Jika Anda menggunakan template untuk menyebarkan VPC Anda sebelum tanggal ini atau Anda menggunakan VPC Anda sendiri, Anda harus mengaktifkan pengaturan ini secara manual. Untuk template, lihat [the section called “Buat VPC”](#). Untuk informasi selengkapnya, lihat [Memodifikasi atribut IPv4 pengalamatan publik untuk subnet Anda](#) dan [Memodifikasi atribut IPv6 pengalamatan untuk subnet Anda di Panduan Pengguna Amazon VPC](#).
- Jika subnet tempat Anda menerapkan node adalah subnet pribadi dan tabel rutanya tidak menyertakan rute ke [perangkat terjemahan alamat jaringan \(NAT\) \(\)](#) atau [gateway khusus egres \(IPv4\)](#) IPv6, tambahkan titik akhir VPC menggunakan ke VPC Anda. AWS PrivateLink Endpoint VPC diperlukan untuk semua AWS layanan yang perlu dikomunikasikan oleh node dan Pod Anda. Contohnya termasuk Amazon ECR, Elastic Load Balancing, CloudWatch Amazon, Security Token Service AWS, dan Amazon Simple Storage Service (Amazon S3). Titik akhir harus menyertakan subnet tempat node berada. Tidak semua AWS layanan mendukung titik akhir VPC. Untuk informasi lebih lanjut, lihat [Apa itu AWS PrivateLink?](#) dan [AWS layanan yang terintegrasi dengan AWS PrivateLink](#). Untuk daftar persyaratan Amazon EKS lainnya, lihat [the section called “kluster privat”](#).
- Jika Anda ingin menerapkan penyeimbang beban ke subnet, subnet harus memiliki tag berikut:
 - Subnet privat

Kunci	Nilai
kubernetes.io/role/internal-elb	1

- Subnet publik

Kunci	Nilai
kubernetes.io/role/elb	1

Ketika klaster Kubernetes yang versi 1.18 dan sebelumnya dibuat, Amazon EKS menambahkan tag berikut ke semua subnet yang ditentukan.

Kunci	Nilai
kubernetes.io/cluster/ <i>my-cluster</i>	shared

Saat Anda membuat cluster Kubernetes baru sekarang, Amazon EKS tidak menambahkan tag ke subnet Anda. Jika tag berada di subnet yang digunakan oleh cluster yang sebelumnya merupakan versi sebelumnya 1.19, tag tidak secara otomatis dihapus dari subnet saat cluster diperbarui ke versi yang lebih baru. Versi 2.1.1 atau sebelumnya dari AWS Load Balancer Controller memerlukan tag ini. Jika Anda menggunakan versi yang lebih baru dari Load Balancer Controller, Anda dapat menghapus tag tanpa mengganggu layanan Anda. Untuk informasi selengkapnya tentang controller, lihat [the section called “AWS Pengontrol Load Balancer”](#).

Jika Anda menerapkan VPC dengan eksctl menggunakan atau salah satu templat AWS CloudFormation VPC Amazon EKS, berikut ini berlaku:

- Pada atau setelah 26 Maret 2020 - IPv4 Alamat publik secara otomatis ditetapkan oleh subnet publik ke node baru yang digunakan untuk subnet publik.
- Sebelum 26 Maret 2020 - IPv4 Alamat publik tidak secara otomatis ditetapkan oleh subnet publik ke node baru yang digunakan ke subnet publik.

Perubahan ini berdampak pada grup node baru yang diterapkan ke subnet publik dengan cara berikut:

- [Grup node terkelola](#) - Jika grup node dikerahkan ke subnet publik pada atau setelah 22 April 2020, penetapan otomatis alamat IP publik harus diaktifkan untuk subnet publik. Untuk informasi selengkapnya, lihat [Memodifikasi atribut IPv4 pengalamatan publik untuk subnet Anda](#).

- Grup node yang dikelola sendiri [Linux](#), [Windows](#), atau [Arm](#) - Jika grup node diterapkan ke subnet publik pada atau setelah 26 Maret 2020, penetapan otomatis alamat IP publik harus diaktifkan untuk subnet publik. Jika tidak, node harus diluncurkan dengan alamat IP publik sebagai gantinya. Untuk informasi selengkapnya, lihat [Memodifikasi atribut IPv4 pengalamatan publik untuk subnet Anda](#) atau [Menetapkan IPv4 alamat publik selama peluncuran](#) instance.

Persyaratan dan pertimbangan subnet bersama

Anda dapat menggunakan berbagi VPC untuk berbagi subnet dengan AWS akun lain dalam Organizations yang sama. AWS Anda dapat membuat klaster Amazon EKS di subnet bersama, dengan pertimbangan berikut:

- Pemilik subnet VPC harus berbagi subnet dengan akun peserta sebelum akun tersebut dapat membuat cluster Amazon EKS di dalamnya.
- Anda tidak dapat meluncurkan sumber daya menggunakan grup keamanan default untuk VPC karena milik pemilik. Selain itu, peserta tidak dapat meluncurkan sumber daya menggunakan grup keamanan yang dimiliki oleh peserta lain atau pemilik.
- Dalam subnet bersama, peserta dan pemilik secara terpisah mengontrol grup keamanan dalam setiap akun masing-masing. Pemilik subnet dapat melihat grup keamanan yang dibuat oleh peserta tetapi tidak dapat melakukan tindakan apa pun pada mereka. Jika pemilik subnet ingin menghapus atau memodifikasi grup keamanan ini, peserta yang membuat grup keamanan harus mengambil tindakan.
- Jika cluster dibuat oleh peserta, pertimbangan berikut berlaku:
 - Peran IAM cluster dan peran IAM Node harus dibuat di akun itu. Untuk informasi selengkapnya, lihat [the section called “IAM role klaster”](#) dan [the section called “IAM role simpul”](#).
 - Semua node harus dibuat oleh peserta yang sama, termasuk grup node terkelola.
- Pemilik VPC bersama tidak dapat melihat, memperbarui, atau menghapus klaster yang dibuat peserta di subnet bersama. Ini merupakan tambahan dari sumber daya VPC yang setiap akun memiliki akses berbeda. Untuk informasi selengkapnya, lihat [Tanggung jawab dan izin untuk pemilik dan peserta](#) di Panduan Pengguna Amazon VPC.
- Jika Anda menggunakan fitur jaringan kustom plugin Amazon VPC CNI untuk Kubernetes, Anda perlu menggunakan pemetaan ID Availability Zone yang tercantum di akun pemilik untuk membuatnya. ENIConfig Untuk informasi selengkapnya, lihat [the section called “Jaringan kustom”](#).

Untuk informasi selengkapnya tentang berbagi subnet VPC, lihat Bagian [VPC Anda dengan akun lain di](#) Panduan Pengguna Amazon VPC.

Buat VPC Amazon untuk kluster Amazon EKS Anda

Anda dapat menggunakan Amazon Virtual Private Cloud (Amazon VPC) untuk meluncurkan AWS sumber daya ke jaringan virtual yang telah Anda tentukan. Jaringan virtual ini sangat mirip dengan jaringan tradisional yang mungkin Anda operasikan di pusat data Anda sendiri. Namun, ia datang dengan manfaat menggunakan infrastruktur yang dapat diskalakan dari Amazon Web Services. Kami menyarankan Anda untuk memiliki pemahaman menyeluruh tentang layanan VPC Amazon sebelum menerapkan cluster Amazon EKS produksi. Untuk informasi selengkapnya, silakan lihat ACL Jaringan di [Panduan Pengguna Amazon VPC](#).

Cluster Amazon EKS, node, dan sumber daya Kubernetes di-deploy ke VPC. Jika Anda ingin menggunakan VPC yang ada dengan Amazon EKS, VPC itu harus memenuhi persyaratan yang dijelaskan di [the section called “Persyaratan VPC dan subnet”](#). Topik ini menjelaskan cara membuat VPC yang memenuhi persyaratan Amazon EKS menggunakan templat yang disediakan AWS CloudFormation Amazon EKS. Setelah menerapkan template, Anda dapat melihat sumber daya yang dibuat oleh template untuk mengetahui dengan tepat sumber daya apa yang dibuatnya, dan konfigurasi sumber daya tersebut. Jika Anda menggunakan node hybrid, VPC Anda harus memiliki rute dalam tabel rute untuk jaringan lokal Anda. Untuk informasi selengkapnya tentang persyaratan jaringan untuk node hybrid, lihat [the section called “Siapkan jaringan”](#).

Prasyarat

Untuk membuat VPC untuk Amazon EKS, Anda harus memiliki izin IAM yang diperlukan untuk membuat sumber daya Amazon VPC. Sumber daya ini adalah VPCs, subnet, grup keamanan, tabel dan rute, dan gateway internet dan NAT. [Untuk informasi selengkapnya, lihat Membuat VPC dengan kebijakan contoh subnet publik di Panduan Pengguna Amazon VPC dan daftar lengkap Tindakan dalam Referensi Otorisasi Layanan.](#)

Anda dapat membuat VPC dengan subnet publik dan privat, subnet publik saja, atau subnet privat saja.

Subnet publik dan pribadi

VPC ini memiliki dua subnet publik dan dua subnet pribadi. Tabel rute terkait subnet publik memiliki rute ke gateway internet. Namun, tabel rute subnet pribadi tidak memiliki rute ke gateway internet.

Satu subnet publik dan satu subnet privat di-deploy ke Availability Zone yang sama. Subnet publik dan swasta lainnya dikerahkan ke Availability Zone kedua di Wilayah yang sama AWS . Kami merekomendasikan opsi ini untuk sebagian besar penerapan.

Dengan opsi ini, Anda dapat menyebarkan node Anda ke subnet pribadi. Opsi ini memungkinkan Kubernetes untuk menyebarkan load balancer ke subnet publik yang dapat memuat lalu lintas keseimbangan ke Pod yang berjalan pada node di subnet pribadi. IPv4Alamat publik secara otomatis ditetapkan ke node yang disebarkan ke subnet publik, tetapi IPv4 alamat publik tidak ditetapkan ke node yang disebarkan ke subnet pribadi.

Anda juga dapat menetapkan IPv6 alamat ke node di subnet publik dan pribadi. Node dalam subnet pribadi dapat berkomunikasi dengan cluster dan AWS layanan lainnya. Pod dapat berkomunikasi ke internet melalui gateway NAT menggunakan IPv4 alamat atau gateway Internet outbound-only menggunakan alamat yang digunakan di setiap Availability IPv6 Zone. Grup keamanan dikerahkan yang memiliki aturan yang menolak semua lalu lintas masuk dari sumber selain cluster atau node tetapi memungkinkan semua lalu lintas keluar. Subnet-subnet ditandai sehingga Kubernetes dapat men-deploy penyeimbang beban ke subnet-subnet tersebut.

- a. Buka [konsol AWS CloudFormation](#) .
- b. Dari bilah navigasi, pilih AWS Wilayah yang mendukung Amazon EKS.
- c. Pilih Buat tumpukan, Dengan sumber daya baru (standar).
- d. Di bawah Prasyarat - Siapkan templat, pastikan Template sudah siap dipilih dan kemudian di bawah Tentukan templat, pilih URL Amazon S3.
- e. Anda dapat membuat VPC yang hanya mendukungIPv4, atau VPC yang mendukung dan. IPv4 IPv6 Tempelkan salah satu dari berikut ini URLs ke area teks di bawah URL Amazon S3 dan pilih Berikutnya:
 - IPv4

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml
```

- IPv4 dan IPv6

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-ipv6-vpc-public-private-subnets.yaml
```


- a. Pada halaman Tentukan detail tumpukan, masukkan parameter, lalu pilih Berikutnya.
 - Nama tumpukan: Pilih nama tumpukan untuk AWS CloudFormation tumpukan Anda. Misalnya, Anda dapat menggunakan nama template yang Anda gunakan pada langkah sebelumnya. Nama hanya dapat berisi karakter alfanumerik (peka huruf besar/kecil) dan tanda hubung. Itu harus dimulai dengan karakter alfanumerik dan tidak boleh lebih dari 100 karakter. Nama harus unik di dalam AWS Wilayah dan AWS akun tempat Anda membuat klaster.
 - VpcBlock: Pilih rentang IPv4 CIDR untuk VPC Anda. Setiap node, Pod, dan load balancer yang Anda gunakan diberi IPv4 alamat dari blok ini. IPv4Nilai default menyediakan alamat IP yang cukup untuk sebagian besar implementasi, tetapi jika tidak, maka Anda dapat mengubahnya. Untuk informasi selengkapnya, lihat [Pengukuran VPC and subnet](#) dalam Panduan Pengguna Amazon VPC. Anda juga dapat menambahkan blok CIDR tambahan ke VPC setelah dibuat. Jika Anda membuat IPv6 VPC, rentang IPv6 CIDR secara otomatis ditetapkan untuk Anda dari ruang Alamat Unicast Global Amazon.
 - PublicSubnet01Block: Tentukan blok IPv4 CIDR untuk subnet publik 1. Nilai default menyediakan alamat IP yang cukup untuk sebagian besar implementasi, tetapi jika tidak, maka Anda dapat mengubahnya. Jika Anda membuat IPv6 VPC, blok ini ditentukan untuk Anda dalam template.
 - PublicSubnet02Block: Tentukan blok IPv4 CIDR untuk subnet publik 2. Nilai default menyediakan alamat IP yang cukup untuk sebagian besar implementasi, tetapi jika tidak, maka Anda dapat mengubahnya. Jika Anda membuat IPv6 VPC, blok ini ditentukan untuk Anda dalam template.
 - PrivateSubnet01Block: Tentukan blok IPv4 CIDR untuk subnet pribadi 1. Nilai default menyediakan alamat IP yang cukup untuk sebagian besar implementasi, tetapi jika tidak, maka Anda dapat mengubahnya. Jika Anda membuat IPv6 VPC, blok ini ditentukan untuk Anda dalam template.
 - PrivateSubnet02Block: Tentukan blok IPv4 CIDR untuk subnet pribadi 2. Nilai default menyediakan alamat IP yang cukup untuk sebagian besar implementasi, tetapi jika tidak, maka Anda dapat mengubahnya. Jika Anda membuat IPv6 VPC, blok ini ditentukan untuk Anda dalam template.
- b. (Opsional) Pada halaman Configure stack options, beri tag sumber daya tumpukan Anda dan kemudian pilih Berikutnya.
- c. Di halaman Tinjau, pilih Buat tumpukan.
- d. Ketika tumpukan Anda dibuat, pilih tumpukan tersebut di konsol dan pilih Outputs.

- e. Rekam VpcId untuk VPC yang dibuat. Anda membutuhkan ini ketika Anda membuat cluster dan node Anda.
- f. Rekam subnet yang dibuat dan apakah Anda membuatnya sebagai subnet publik atau pribadi. SubnetIds Anda memerlukan setidaknya dua dari ini ketika Anda membuat cluster dan node Anda.
- g. Jika Anda membuat IPv4 VPC, lewati langkah ini. Jika Anda membuat IPv6 VPC, Anda harus mengaktifkan opsi IPv6 alamat penetapan otomatis untuk subnet publik yang dibuat oleh templat. Pengaturan itu sudah diaktifkan untuk subnet pribadi. Untuk mengaktifkan pengaturan, selesaikan langkah-langkah berikut:
 - i. Buka konsol Amazon VPC di <https://console.aws.amazon.com/vpc/>
 - ii. Di panel navigasi kiri, pilih Subnet
 - iii. Pilih salah satu subnet publik Anda (**stack-name**/SubnetPublic01 atau **stack-name**/SubnetPublic02 berisi kata publik) dan pilih Tindakan, Edit pengaturan subnet.
 - iv. Pilih kotak centang Aktifkan IPv6 alamat penetapan otomatis dan kemudian pilih Simpan.
 - v. Selesaikan langkah-langkah sebelumnya lagi untuk subnet publik Anda yang lain.

Hanya subnet publik

VPC ini memiliki tiga subnet publik yang dikerahkan ke Availability Zone yang berbeda di suatu Wilayah. AWS Semua node secara otomatis diberi IPv4 alamat publik dan dapat mengirim dan menerima lalu lintas internet melalui [gateway internet](#). Sebuah [grup subnet](#) di-deploy, dimana ia menolak semua lalu lintas masuk dan mengizinkan semua lalu lintas keluar. Subnet-subnet ditandai sehingga Kubernetes dapat men-deploy penyeimbang beban ke subnet-subnet tersebut.

- a. Buka [konsol AWS CloudFormation](#) .
- b. Dari bilah navigasi, pilih AWS Wilayah yang mendukung Amazon EKS.
- c. Pilih Buat tumpukan, Dengan sumber daya baru (standar).
- d. Di bawah Siapkan template, pastikan Template sudah siap dipilih dan kemudian di bawah Sumber template, pilih URL Amazon S3.
- e. Rekatkan URL berikut ke area teks di bawah URL Amazon S3 dan pilih Berikutnya:

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-vpc-sample.yaml
```

- a. Pada halaman Tentukan Detail, masukkan parameter, lalu pilih Berikutnya.

- Nama tumpukan: Pilih nama tumpukan untuk AWS CloudFormation tumpukan Anda. Misalnya, Anda bisa menyebutnya *amazon-eks-vpc-sample*. Nama hanya dapat berisi karakter alfanumerik (peka huruf besar/kecil) dan tanda hubung. Itu harus dimulai dengan karakter alfanumerik dan tidak boleh lebih dari 100 karakter. Nama harus unik di dalam AWS Wilayah dan AWS akun tempat Anda membuat klaster.
 - VpcBlock: Pilih blok CIDR untuk VPC Anda. Setiap node, Pod, dan load balancer yang Anda gunakan diberi IPv4 alamat dari blok ini. IPv4 Nilai default menyediakan alamat IP yang cukup untuk sebagian besar implementasi, tetapi jika tidak, maka Anda dapat mengubahnya. Untuk informasi selengkapnya, lihat [Pengukuran VPC and subnet](#) dalam Panduan Pengguna Amazon VPC. Anda juga dapat menambahkan blok CIDR tambahan ke VPC setelah dibuat.
 - Subnet01Block: Tentukan blok CIDR untuk subnet 1. Nilai default menyediakan alamat IP yang cukup untuk sebagian besar implementasi, tetapi jika tidak, maka Anda dapat mengubahnya.
 - Subnet02Block: Tentukan blok CIDR untuk subnet 2. Nilai default menyediakan alamat IP yang cukup untuk sebagian besar implementasi, tetapi jika tidak, maka Anda dapat mengubahnya.
 - Subnet03Block: Tentukan blok CIDR untuk subnet 3. Nilai default menyediakan alamat IP yang cukup untuk sebagian besar implementasi, tetapi jika tidak, maka Anda dapat mengubahnya.
- b. (Opsional) Di halaman Opsi, tandai sumber daya tumpukan Anda. Pilih Berikutnya.
- c. Pada halaman Tinjauan, pilih Buat.
- d. Ketika tumpukan Anda dibuat, pilih tumpukan tersebut di konsol dan pilih Outputs.
- e. Rekam VpcId untuk VPC yang dibuat. Anda membutuhkan ini ketika Anda membuat cluster dan node Anda.
- f. Rekam SubnetIds untuk subnet yang dibuat. Anda memerlukan setidaknya dua dari ini ketika Anda membuat cluster dan node Anda.
- g. (Opsional) Kluster apa pun yang Anda terapkan ke VPC ini dapat menetapkan alamat IPv4 pribadi ke Pod dan layanan Anda. Jika Anda ingin menerapkan cluster ke VPC ini untuk menetapkan IPv6 alamat pribadi ke Pod dan layanan Anda, lakukan pembaruan pada VPC, subnet, tabel rute, dan grup keamanan Anda. Untuk informasi selengkapnya, lihat [Memigrasi yang ada VPCs dari IPv4 ke IPv6](#) dalam Panduan Pengguna Amazon VPC. Amazon EKS mengharuskan subnet Anda mengaktifkan opsi Auto-assign IPv6 alamat. Secara default, ini dinonaktifkan.

Hanya subnet pribadi

VPC ini memiliki tiga subnet pribadi yang digunakan ke Availability Zone yang berbeda di Region. AWS Sumber daya yang dikerahkan ke subnet tidak dapat mengakses internet, juga tidak dapat mengakses sumber daya internet di subnet. Template membuat [titik akhir VPC](#) menggunakan AWS PrivateLink beberapa AWS layanan yang biasanya perlu diakses oleh node. Jika node Anda memerlukan akses internet keluar, Anda dapat menambahkan [gateway NAT publik di](#) Availability Zone setiap subnet setelah VPC dibuat. [Grup keamanan](#) dibuat yang menyangkal semua lalu lintas masuk, kecuali dari sumber daya yang dikerahkan ke subnet. Grup keamanan juga memungkinkan semua lalu lintas keluar. Subnet-subnet ditandai sehingga Kubernetes dapat mendeploy penyeimbang beban internal ke subnet-subnet tersebut. Jika Anda membuat VPC dengan konfigurasi ini, lihat [the section called “klaster privat”](#) persyaratan dan pertimbangan tambahan.

- a. Buka [konsol AWS CloudFormation](#).
- b. Dari bilah navigasi, pilih AWS Wilayah yang mendukung Amazon EKS.
- c. Pilih Buat tumpukan, Dengan sumber daya baru (standar).
- d. Di bawah Siapkan template, pastikan Template sudah siap dipilih dan kemudian di bawah Sumber template, pilih URL Amazon S3.
- e. Rekatkan URL berikut ke area teks di bawah URL Amazon S3 dan pilih Berikutnya:

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-fully-private-vpc.yaml
```

- a. Pada halaman Tentukan Detail, masukkan parameter dan kemudian pilih Berikutnya.
 - Nama tumpukan: Pilih nama tumpukan untuk AWS CloudFormation tumpukan Anda. Misalnya, Anda bisa menyebutnya *amazon-eks-fully-private-vpc*. Nama hanya dapat berisi karakter alfanumerik (peka huruf besar/kecil) dan tanda hubung. Itu harus dimulai dengan karakter alfanumerik dan tidak boleh lebih dari 100 karakter. Nama harus unik di dalam AWS Wilayah dan AWS akun tempat Anda membuat klaster.
 - VpcBlock: Pilih blok CIDR untuk VPC Anda. Setiap node, Pod, dan load balancer yang Anda gunakan diberi IPv4 alamat dari blok ini. IPv4Nilai default menyediakan alamat IP yang cukup untuk sebagian besar implementasi, tetapi jika tidak, maka Anda dapat mengubahnya. Untuk informasi selengkapnya, lihat [Pengukuran VPC and subnet](#) dalam Panduan Pengguna Amazon VPC. Anda juga dapat menambahkan blok CIDR tambahan ke VPC setelah dibuat.

- PrivateSubnet01Block: Tentukan blok CIDR untuk subnet 1. Nilai default menyediakan alamat IP yang cukup untuk sebagian besar implementasi, tetapi jika tidak, maka Anda dapat mengubahnya.
 - PrivateSubnet02Block: Tentukan blok CIDR untuk subnet 2. Nilai default menyediakan alamat IP yang cukup untuk sebagian besar implementasi, tetapi jika tidak, maka Anda dapat mengubahnya.
 - PrivateSubnet03Block: Tentukan blok CIDR untuk subnet 3. Nilai default menyediakan alamat IP yang cukup untuk sebagian besar implementasi, tetapi jika tidak, maka Anda dapat mengubahnya.
- b. (Opsional) Di halaman Opsi, tandai sumber daya tumpukan Anda. Pilih Berikutnya.
- c. Pada halaman Tinjauan, pilih Buat.
- d. Ketika tumpukan Anda dibuat, pilih tumpukan tersebut di konsol dan pilih Outputs.
- e. Rekam VpcId untuk VPC yang dibuat. Anda membutuhkan ini ketika Anda membuat cluster dan node Anda.
- f. Rekam SubnetIds untuk subnet yang dibuat. Anda memerlukan setidaknya dua dari ini ketika Anda membuat cluster dan node Anda.
- g. (Opsional) Kluster apa pun yang Anda terapkan ke VPC ini dapat menetapkan alamat IPv4 pribadi ke Pod dan layanan Anda. Jika Anda ingin menerapkan cluster ke VPC ini untuk menetapkan IPv6 alamat pribadi ke Pod dan layanan Anda, lakukan pembaruan pada VPC, subnet, tabel rute, dan grup keamanan Anda. Untuk informasi selengkapnya, lihat [Memigrasi yang ada VPCs dari IPv4 ke IPv6](#) dalam Panduan Pengguna Amazon VPC. Amazon EKS mengharuskan subnet Anda mengaktifkan opsi Auto-assign IPv6 alamat (dinonaktifkan secara default).

Lihat persyaratan grup keamanan Amazon EKS untuk cluster

Topik ini menjelaskan persyaratan grup keamanan klaster Amazon EKS.

Grup keamanan klaster default

Saat Anda membuat klaster, Amazon EKS membuat grup keamanan yang diberi nama `eks-cluster-sg-my-cluster-uniqueID`. Grup keamanan ini memiliki aturan default berikut:

Jenis aturan	Protokol	Port	Sumber	Tujuan
Ke dalam	Semua	Semua	Diri Sendiri	
Ke luar	Semua	Semua		0.0.0.0/0 (IPv4) atau: :/0 () IPv6
Ke luar	Semua	Semua		Mandiri (untuk lalu lintas EFA)

Grup keamanan default menyertakan aturan keluar yang memungkinkan lalu lintas Elastic Fabric Adapter (EFA) dengan tujuan grup keamanan yang sama. Hal ini memungkinkan lalu lintas EFA dalam cluster, yang bermanfaat untuk beban kerja AI/ML dan High Performance Computing (HPC). Untuk informasi selengkapnya, lihat [Adaptor Kain Elastis untuk beban kerja AI/ML dan HPC di Amazon EC2 di Panduan Pengguna Amazon](#) Elastic Compute Cloud.

Important

Jika klaster Anda tidak memerlukan aturan keluar, Anda dapat menghapusnya. Jika Anda menghapusnya, Anda masih harus memiliki aturan minimum yang tercantum dalam [Membatasi lalu lintas cluster](#). Jika Anda menghapus aturan masuk, Amazon EKS akan membuatnya ulang setiap kali cluster diperbarui.

Amazon EKS menambahkan tag berikut ke grup keamanan. Jika Anda menghapus tag, Amazon EKS menemukannya kembali ke grup keamanan setiap kali klaster Anda diperbarui.

Kunci	Nilai
kubernetes.io/cluster/ <i>my-cluster</i>	owned
aws:eks:cluster-name	<i>my-cluster</i>
Name	eks-cluster-sg- <i>my-cluste</i> <i>r -uniqueid</i>

Amazon EKS secara otomatis mengaitkan grup keamanan ini ke sumber daya berikut yang juga dibuatnya:

- 2—4 antarmuka jaringan elastis (disebut untuk sisa dokumen ini sebagai antarmuka jaringan) yang dibuat saat Anda membuat cluster Anda.
- Antarmuka jaringan node dalam grup node terkelola apa pun yang Anda buat.

Aturan default memungkinkan semua lalu lintas mengalir bebas antara cluster dan node Anda, dan memungkinkan semua lalu lintas keluar ke tujuan mana pun. Saat membuat kluster, Anda dapat (opsional) menentukan grup keamanan Anda sendiri. Jika ya, Amazon EKS juga mengaitkan grup keamanan yang Anda tentukan ke antarmuka jaringan yang dibuatnya untuk cluster Anda. Namun, itu tidak mengaitkannya dengan grup simpul apa pun yang Anda buat.

Anda dapat menentukan ID grup keamanan kluster Anda di AWS Management Console bawah bagian Jaringan kluster. Atau, Anda dapat melakukannya dengan menjalankan perintah AWS CLI berikut.

```
aws eks describe-cluster --name my-cluster --query  
cluster.resourcesVpcConfig.clusterSecurityGroupId
```

Membatasi lalu lintas kluster

Jika Anda perlu membatasi port terbuka antara cluster dan node, Anda dapat menghapus [aturan keluar default](#) dan menambahkan aturan minimum berikut yang diperlukan untuk cluster. Jika Anda menghapus [aturan masuk default](#), Amazon EKS akan membuatnya ulang setiap kali cluster diperbarui.

Jenis aturan	Protokol	Port	Tujuan
Ke luar	TCP	443	Grup keamanan kluster
Ke luar	TCP	10250	Grup keamanan kluster
Keluar (DNS)	TCP dan UDP	53	Grup keamanan kluster

Anda juga harus menambahkan aturan untuk lalu lintas berikut:

- Protokol dan port apa pun yang Anda harapkan untuk digunakan node untuk komunikasi antar simpul.
- Akses internet keluar sehingga node dapat mengakses Amazon EKS APIs untuk introspeksi cluster dan pendaftaran node pada waktu peluncuran. Jika node Anda tidak memiliki akses internet, tinjau [Deploy private cluster dengan akses internet terbatas untuk pertimbangan](#) tambahan.
- Akses node untuk menarik gambar kontainer dari Amazon ECR atau pendaftar kontainer lain APIs yang mereka butuhkan untuk menarik gambar, seperti DockerHub Untuk informasi selengkapnya, lihat [rentang alamat AWS IP](#) di Referensi AWS Umum.
- Akses node ke Amazon S3.
- Aturan terpisah diperlukan untuk IPv4 dan IPv6 alamat.
- Jika Anda menggunakan node hibrid, Anda harus menambahkan grup keamanan tambahan ke klaster Anda untuk memungkinkan komunikasi dengan node dan pod lokal Anda. Untuk informasi selengkapnya, lihat [the section called “Siapkan jaringan”](#).

Jika Anda mempertimbangkan untuk membatasi aturan, sebaiknya Anda menguji semua Pod secara menyeluruh sebelum menerapkan aturan yang diubah ke klaster produksi.

Jika Anda awalnya menggunakan klaster dengan Kubernetes 1.14 dan versi platform eks.3 atau sebelumnya, pertimbangkan hal berikut:

- Anda mungkin juga memiliki control plane dan grup keamanan node. Ketika grup ini dibuat, mereka menyertakan aturan terbatas yang tercantum dalam tabel sebelumnya. Grup keamanan ini tidak lagi diperlukan dan dapat dihapus. Namun, Anda perlu memastikan grup keamanan klaster Anda berisi aturan yang dikandung grup tersebut.
- Jika Anda menerapkan cluster menggunakan API secara langsung atau Anda menggunakan alat seperti AWS CLI AWS CloudFormation atau untuk membuat cluster dan Anda tidak menentukan grup keamanan pada pembuatan klaster, maka grup keamanan default untuk VPC diterapkan ke antarmuka jaringan cluster yang dibuat Amazon EKS.

Grup keamanan bersama

Amazon EKS mendukung grup keamanan bersama.

- Asosiasi VPC Grup Keamanan mengaitkan grup keamanan dengan beberapa VPCs di akun dan wilayah yang sama.
- Pelajari cara [Mengaitkan grup keamanan dengan beberapa VPCs](#) di Panduan Pengguna Amazon VPC.
- Grup keamanan bersama memungkinkan Anda berbagi grup keamanan dengan AWS akun lain. Akun harus berada di AWS organisasi yang sama.
- Pelajari cara [Berbagi grup keamanan dengan organisasi](#) di Panduan Pengguna Amazon VPC.
- Kelompok keamanan selalu terbatas pada satu AWS wilayah.

Pertimbangan untuk Amazon EKS

- EKS memiliki persyaratan yang sama untuk grup keamanan bersama atau multi-VPC sebagai grup keamanan standar.

Kelola add-on jaringan untuk kluster Amazon EKS

Beberapa add-on jaringan tersedia untuk kluster Amazon EKS Anda.

Pengaya bawaan

Note

Saat Anda membuat kluster EKS:

- Menggunakan AWS Konsol: Add-on bawaan (seperti CoreDNS, kube-proxy, dll.) Secara otomatis diinstal sebagai Add-on Amazon EKS. Ini dapat dengan mudah dikonfigurasi dan diperbarui melalui AWS Konsol, CLI, atau SDKs
- Menggunakan metode lain (CLI, SDKs, dll.): Add-on bawaan yang sama diinstal sebagai versi yang dikelola sendiri yang berjalan sebagai penerapan Kubernetes biasa. Ini memerlukan konfigurasi dan pembaruan manual karena tidak dapat dikelola melalui AWS alat.

Sebaiknya gunakan Add-on Amazon EKS daripada versi yang dikelola sendiri untuk menyederhanakan pengelolaan add-on dan mengaktifkan konfigurasi dan pembaruan terpusat melalui layanan. AWS

Plugin Amazon VPC CNI untuk Kubernetes

Add-on CNI ini menciptakan antarmuka jaringan elastis dan menempelkannya ke node Amazon Anda. EC2 Add-on juga memberikan IPv6 alamat pribadi IPv4 atau dari VPC Anda ke setiap Pod dan layanan. Add-on ini diinstal, secara default, di cluster Anda. Untuk informasi selengkapnya, lihat [the section called “Amazon VPC CNI”](#). Jika Anda menggunakan node hybrid, VPC CNI masih diinstal secara default tetapi dicegah berjalan pada node hybrid Anda dengan aturan anti-afinitas. Untuk informasi selengkapnya tentang opsi CNI Anda untuk node hibrida, lihat [the section called “Konfigurasi CNI”](#).

CoreDNS

CoreDNS adalah server DNS yang fleksibel dan dapat diperluas yang dapat berfungsi sebagai DNS kluster Kubernetes. CoreDNS menyediakan resolusi nama untuk semua Pod di cluster. Add-on ini diinstal, secara default, di cluster Anda. Untuk informasi selengkapnya, lihat [the section called “CoreDNS”](#).

kube-proxy

Add-on ini mempertahankan aturan jaringan pada EC2 node Amazon Anda dan memungkinkan komunikasi jaringan ke Pod Anda. Add-on ini diinstal, secara default, di cluster Anda. Untuk informasi selengkapnya, lihat [the section called “kube-proxy”](#).

Pengaya AWS jaringan opsional

AWS Pengontrol Load Balancer

Ketika Anda menerapkan objek layanan Kubernetes dari tipe `loadbalancer`, controller akan membuat AWS Network Load Balancers. Saat Anda membuat objek ingress Kubernetes, controller akan membuat AWS Application Load Balancers. Sebaiknya gunakan pengontrol ini untuk menyediakan Network Load Balancers, daripada menggunakan pengontrol [Cloud Provider lama](#) yang ada di Kubernetes. Untuk informasi selengkapnya, lihat dokumentasi [AWS Load Balancer Controller](#).

AWS Pengontrol API Gerbang

[Kontroler ini memungkinkan Anda menghubungkan layanan di beberapa kluster Kubernetes menggunakan API gateway Kubernetes. Kontroler menghubungkan layanan Kubernetes yang berjalan di EC2 instans Amazon, container, dan fungsi tanpa server dengan menggunakan layanan Amazon VPC Lattice.](#) Untuk informasi selengkapnya, lihat dokumentasi [AWS Gateway API Controller](#).

Untuk informasi selengkapnya tentang add-on, lihat [the section called “Add-on Amazon EKS”](#).

Tetapkan IPs ke Pod dengan Amazon VPC CNI

Tip

Dengan Amazon EKS Auto Mode, Anda tidak perlu menginstal atau meningkatkan add-on jaringan. Mode Otomatis mencakup jaringan pod dan kemampuan load balancing. Untuk informasi selengkapnya, lihat [Mode Otomatis EKS](#).

Plugin Amazon VPC CNI untuk add-on Kubernetes diterapkan di setiap node Amazon di cluster Amazon EC2 EKS Anda. Add-on membuat [antarmuka jaringan elastis](#) dan menempelkannya ke node Amazon Anda. EC2 Add-on juga memberikan IPv6 alamat pribadi IPv4 atau dari VPC Anda ke setiap Pod.

Versi add-on digunakan dengan setiap node Fargate di cluster Anda, tetapi Anda tidak memperbaruinya di node Fargate. Plugin CNI lain yang kompatibel tersedia untuk digunakan pada kluster Amazon EKS, tetapi ini adalah satu-satunya plugin CNI yang didukung oleh Amazon EKS untuk node yang berjalan pada infrastruktur. AWS Untuk informasi selengkapnya tentang plugin CNI lain yang kompatibel, lihat. [the section called “Plugin CNI alternatif”](#) VPC CNI tidak didukung untuk digunakan dengan node hybrid. Untuk informasi selengkapnya tentang opsi CNI Anda untuk node hibrida, lihat [the section called “Konfigurasi CNI”](#).

Tabel berikut mencantumkan versi terbaru yang tersedia dari jenis add-on Amazon EKS untuk setiap versi Kubernetes.

Versi Amazon VPC CNI

Versi Kubernetes	Amazon EKS jenis versi VPC CNI
1.33	v1.19.6-eksbuild.1
1.32	v1.19.6-eksbuild.1
1.31	v1.19.6-eksbuild.1
1.30	v1.19.6-eksbuild.1

Versi Kubernetes	Amazon EKS jenis versi VPC CNI
1.29	v1.19.6-eksbuild.1
1.28	v1.19.6-eksbuild.1
1.27	v1.19.6-eksbuild.1
1.26	v1.19.6-eksbuild.1

Important

Jika Anda mengelola sendiri add-on ini, versi dalam tabel mungkin tidak sama dengan versi yang dikelola sendiri yang tersedia. Untuk informasi selengkapnya tentang memperbarui jenis pengaya yang dikelola sendiri, lihat. [the section called “Pembaruan \(dikelola sendiri\)”](#)

Important

Untuk meningkatkan ke VPC CNI v1.12.0 atau yang lebih baru, Anda harus meng-upgrade ke VPC CNI v1.7.0 terlebih dahulu. Kami menyarankan Anda memperbarui satu versi minor sekaligus.

Pertimbangan

Berikut ini adalah pertimbangan untuk menggunakan fitur ini.

- Versi ditentukan sebagai `major-version.minor-version.patch-version-eksbuild.build-number`.
- Periksa kompatibilitas versi untuk setiap fitur. Beberapa fitur dari setiap rilis plugin Amazon VPC CNI untuk Kubernetes memerlukan versi Kubernetes tertentu. Saat menggunakan fitur Amazon EKS yang berbeda, jika versi add-on tertentu diperlukan, maka itu dicatat dalam dokumentasi fitur. Kecuali Anda memiliki alasan khusus untuk menjalankan versi sebelumnya, kami sarankan untuk menjalankan versi terbaru.

Buat Amazon VPC CNI (add-on Amazon EKS)

Gunakan langkah-langkah berikut untuk membuat plugin Amazon VPC CNI untuk add-on Kubernetes Amazon EKS.

Sebelum Anda mulai, tinjau pertimbangannya. Untuk informasi selengkapnya, lihat [the section called “Pertimbangan”](#).

Prasyarat

Berikut ini adalah prasyarat untuk plugin Amazon VPC CNI untuk add-on Kubernetes Amazon EKS.

- Sebuah klaster Amazon EKS yang sudah ada. Untuk menyebarkan satu, lihat [Memulai](#).
- Penyedia OpenID Connect (OIDC) AWS Identity and Access Management (IAM) yang sudah ada untuk klaster Anda. Untuk menentukan apakah Anda sudah memiliki satu, atau harus membuat satu, lihat [the section called “Penyedia IAM OIDC”](#).
- Peran IAM dengan kebijakan IAM [AmazonEKS_CNI_Policy](#) (jika klaster Anda menggunakan keluarga) atau kebijakan (jika klaster Anda menggunakan keluarga) IPv4 yang dilampirkan padanya. IPv6 Untuk informasi selengkapnya tentang peran VPC CNI, lihat [the section called “Konfigurasi untuk IRSA”](#) Untuk informasi tentang IPv6 kebijakan, lihat [the section called “Buat kebijakan IAM untuk cluster yang menggunakan keluarga IPv6”](#).

Important

Plugin Amazon VPC CNI untuk versi Kubernetes untuk mengimplementasikan versi v1.16.0 spesifikasi CNI. v1.16.1 v1.0.0 Untuk informasi selengkapnya v1.0.0 tentang spesifikasi CNI, lihat Spesifikasi [Container Network Interface \(CNI\)](#) di. GitHub

Prosedur

Setelah Anda menyelesaikan prasyarat, gunakan langkah-langkah berikut untuk membuat add-on.

1. Lihat versi add-on mana yang diinstal pada cluster Anda.

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |  
cut -d : -f 3
```

Contoh output adalah sebagai berikut.

```
v1.16.4-eksbuild.2
```

2. Lihat jenis add-on yang diinstal pada cluster Anda. Bergantung pada alat yang digunakan untuk membuat kluster, saat ini Anda mungkin tidak menginstal jenis add-on Amazon EKS di kluster Anda. Ganti *my-cluster* dengan nama kluster Anda.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query  
addon.addonVersion --output text
```

Jika nomor versi dikembalikan, Anda memiliki jenis add-on Amazon EKS yang diinstal pada cluster Anda dan tidak perlu menyelesaikan langkah-langkah yang tersisa dalam prosedur ini. Jika kesalahan dikembalikan, Anda tidak memiliki jenis add-on Amazon EKS yang diinstal pada cluster Anda. Selesaikan langkah-langkah yang tersisa dari prosedur ini untuk menginstalnya.

3. Simpan konfigurasi add-on yang Anda instal saat ini.

```
kubectl get daemonset aws-node -n kube-system -o yaml > aws-k8s-cni-old.yaml
```

4. Buat add-on menggunakan AWS CLI. Jika Anda ingin menggunakan AWS Management Console atau `eksctl` untuk membuat add-on, lihat [the section called “Buat add-on”](#) dan tentukan nama `vpc-cni` add-on. Salin perintah yang mengikuti ke perangkat Anda. Buat modifikasi berikut pada perintah, sesuai kebutuhan, dan kemudian jalankan perintah yang dimodifikasi.

- Ganti *my-cluster* dengan nama kluster Anda.
- Ganti *v1.19.6-eksbuild.1* dengan versi terbaru yang tercantum dalam tabel versi terbaru untuk versi cluster Anda. Untuk tabel versi terbaru, lihat [the section called “Versi Amazon VPC CNI”](#).
- Ganti *111122223333* dengan ID akun Anda dan *AmazonEKSVPCCNIRole* dengan nama [peran IAM yang ada](#) yang telah Anda buat. Menentukan peran mengharuskan Anda memiliki penyedia IAM OpenID Connect (OIDC) untuk kluster Anda. Untuk menentukan apakah Anda memiliki satu untuk cluster Anda, atau untuk membuatnya, lihat [the section called “Penyedia IAM OIDC”](#).

```
aws eks create-addon --cluster-name my-cluster --addon-name vpc-cni --addon-version  
v1.19.6-eksbuild.1 \  
--service-account-role-arn arn:aws:iam::111122223333:role/AmazonEKSVPCCNIRole
```

Jika Anda telah menerapkan pengaturan khusus ke add-on saat ini yang bertentangan dengan pengaturan default add-on Amazon EKS, pembuatan mungkin gagal. Jika pembuatan gagal,

Anda menerima kesalahan yang dapat membantu Anda menyelesaikan masalah. Atau, Anda dapat menambahkan `--resolve-conflicts OVERWRITE` ke perintah sebelumnya. Hal ini memungkinkan add-on untuk menimpa pengaturan kustom yang ada. Setelah Anda membuat add-on, Anda dapat memperbaruinya dengan pengaturan khusus Anda.

5. Konfirmasikan bahwa versi terbaru dari add-on untuk versi Kubernetes klaster Anda telah ditambahkan ke klaster Anda. Ganti *my-cluster* dengan nama klaster Anda.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query  
addon.addonVersion --output text
```

Mungkin perlu beberapa detik untuk menyelesaikan pembuatan add-on.

Contoh output adalah sebagai berikut.

```
v1.19.6-eksbuild.1
```

6. Jika Anda membuat pengaturan khusus untuk add-on asli Anda, sebelum Anda membuat add-on Amazon EKS, gunakan konfigurasi yang Anda simpan di langkah sebelumnya untuk memperbarui add-on EKS dengan pengaturan khusus Anda. Ikuti langkah-langkah di [the section called “Update \(EKS add-on\)”](#).
7. (Opsional) Instal `cni-metrics-helper` ke cluster Anda. Ini mengikis elastic network interface dan informasi alamat IP, menggabungkannya pada tingkat cluster, dan menerbitkan metrik ke Amazon CloudWatch. Untuk informasi lebih lanjut, lihat [cni-metrics-helper](#) di GitHub.

Perbarui Amazon VPC CNI (add-on Amazon EKS)

Perbarui jenis Amazon EKS dari plugin Amazon VPC CNI untuk add-on Kubernetes. Jika Anda belum menambahkan jenis add-on Amazon EKS ke cluster Anda, Anda dapat menginstalnya dengan mengikuti [the section called “Buat”](#). Atau, perbarui jenis instalasi VPC CNI lainnya dengan mengikuti [the section called “Pembaruan \(dikelola sendiri\)”](#)

1. Lihat versi add-on mana yang diinstal pada cluster Anda. Ganti *my-cluster* dengan nama klaster Anda.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query  
"addon.addonVersion" --output text
```

Contoh output adalah sebagai berikut.

```
v1.19.6-eksbuild.1
```

Bandingkan versi dengan tabel versi terbaru di [the section called “Versi Amazon VPC CNI”](#). Jika versi yang dikembalikan sama dengan versi untuk versi Kubernetes cluster Anda di tabel versi terbaru, maka Anda sudah menginstal versi terbaru di cluster Anda dan tidak perlu menyelesaikan sisa prosedur ini. Jika Anda menerima kesalahan, alih-alih nomor versi dalam output Anda, maka Anda tidak memiliki jenis add-on Amazon EKS yang diinstal pada cluster Anda. Anda perlu membuat add-on sebelum Anda dapat memperbaruinya dengan prosedur ini. Untuk membuat jenis Amazon EKS dari add-on VPC CNI, Anda dapat mengikuti [the section called “Buat”](#)

2. Simpan konfigurasi add-on yang Anda instal saat ini.

```
kubectl get daemonset aws-node -n kube-system -o yaml > aws-k8s-cni-old.yaml
```

3. Perbarui add-on Anda menggunakan AWS CLI. Jika Anda ingin menggunakan AWS Management Console atau `eksctl` memperbarui add-on, lihat [the section called “Perbarui add-on”](#). Salin perintah yang mengikuti ke perangkat Anda. Buat modifikasi berikut pada perintah, sesuai kebutuhan, dan kemudian jalankan perintah yang dimodifikasi.

- Ganti *my-cluster* dengan nama klaster Anda.
- Ganti *v1.19.6-eksbuild.1* dengan versi terbaru yang tercantum dalam tabel versi terbaru untuk versi cluster Anda.
- Ganti *111122223333* dengan ID akun Anda dan *AmazonEKSVPCCNIRole* dengan nama peran IAM yang ada yang telah Anda buat. Untuk membuat peran IAM untuk VPC CNI, lihat [the section called “Langkah 1: Buat plugin Amazon VPC CNI untuk peran IAM Kubernetes”](#). Menentukan peran mengharuskan Anda memiliki penyedia IAM OpenID Connect (OIDC) untuk klaster Anda. Untuk menentukan apakah Anda memiliki satu untuk cluster Anda, atau untuk membuatnya, lihat [the section called “Penyedia IAM OIDC”](#).
- `--resolve-conflicts PRESERVE`Opsi ini mempertahankan nilai konfigurasi yang ada untuk add-on. Jika Anda telah menetapkan nilai kustom untuk pengaturan add-on, dan Anda tidak menggunakan opsi ini, Amazon EKS menimpa nilai Anda dengan nilai defaultnya. Jika Anda menggunakan opsi ini, kami sarankan untuk menguji setiap bidang dan perubahan nilai pada klaster non-produksi sebelum memperbarui add-on pada cluster produksi Anda. Jika Anda mengubah nilai ini menjadi `OVERWRITE`, semua pengaturan diubah ke nilai default Amazon EKS. Jika Anda telah menetapkan nilai kustom untuk setelan apa pun, nilai tersebut mungkin akan ditimpa dengan nilai default Amazon EKS. Jika Anda mengubah nilai ini menjadi `none`, Amazon EKS

tidak mengubah nilai pengaturan apa pun, tetapi pembaruan mungkin gagal. Jika pembaruan gagal, Anda menerima pesan galat untuk membantu menyelesaikan konflik.

- Jika Anda tidak memperbarui pengaturan konfigurasi, hapus `--configuration-values '{"env":{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}}'` dari perintah. Jika Anda memperbarui pengaturan konfigurasi, ganti `"env":{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}` dengan pengaturan yang ingin Anda atur. Dalam contoh ini, variabel `AWS_VPC_K8S_CNI_EXTERNALSNAT` lingkungan diatur ke `true`. Nilai yang Anda tentukan harus valid untuk skema konfigurasi. Jika Anda tidak tahu skema konfigurasi, jalankan `aws eks describe-addon-configuration --addon-name vpc-cni --addon-version v1.19.6-eksbuild.1`, ganti `v1.19.6-eksbuild.1` dengan nomor versi add-on yang ingin Anda lihat konfigurasinya. Skema dikembalikan dalam output. Jika Anda memiliki konfigurasi kustom yang ada, ingin menghapus semuanya, dan mengatur nilai untuk semua pengaturan kembali ke default Amazon EKS, hapus `"env":{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}` dari perintah, sehingga Anda kosong. `{}` Untuk penjelasan tentang setiap pengaturan, lihat [Variabel Konfigurasi CNI](#) pada GitHub.

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-version
v1.19.6-eksbuild.1 \
    --service-account-role-arn arn:aws:iam::111122223333:role/AmazonEKSVPCCNIRole
\
    --resolve-conflicts PRESERVE --configuration-values '{"env":
{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}}'
```

Mungkin perlu beberapa detik untuk pembaruan selesai.

4. Konfirmasikan bahwa versi add-on telah diperbarui. Ganti `my-cluster` dengan nama kluster Anda.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni
```

Mungkin perlu beberapa detik untuk pembaruan selesai.

Contoh output adalah sebagai berikut.

```
{
  "addon": {
    "addonName": "vpc-cni",
    "clusterName": "my-cluster",
    "status": "ACTIVE",
```

```

    "addonVersion": "v1.19.6-eksbuild.1",
    "health": {
      "issues": []
    },
    "addonArn": "arn:aws:eks:region:111122223333:addon/my-cluster/vpc-
cni/74c33d2f-b4dc-8718-56e7-9fdfa65d14a9",
    "createdAt": "2023-04-12T18:25:19.319000+00:00",
    "modifiedAt": "2023-04-12T18:40:28.683000+00:00",
    "serviceAccountRoleArn": "arn:aws:iam::111122223333:role/
AmazonEKSVPCCNIRole",
    "tags": {},
    "configurationValues": "{\"env\":{\"AWS_VPC_K8S_CNI_EXTERNALSNAT\": \"true
\"}}\"
  }
}

```

Perbarui Amazon VPC CNI (add-on yang dikelola sendiri)

Important

Sebaiknya tambahkan jenis add-on Amazon EKS ke kluster Anda alih-alih menggunakan jenis add-on yang dikelola sendiri. Jika Anda tidak terbiasa dengan perbedaan antara jenis, lihat [the section called “Add-on Amazon EKS”](#). Untuk informasi selengkapnya tentang menambahkan add-on Amazon EKS ke kluster Anda, lihat [the section called “Buat add-on”](#). Jika Anda tidak dapat menggunakan add-on Amazon EKS, kami mendorong Anda untuk mengirimkan masalah tentang mengapa Anda tidak dapat ke repositori [peta jalan GitHub Containers](#).

1. Konfirmasikan bahwa Anda tidak memiliki jenis add-on Amazon EKS yang diinstal pada cluster Anda. Ganti *my-cluster* dengan nama kluster Anda.

```

aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
addon.addonVersion --output text

```

Jika pesan kesalahan dikembalikan, Anda tidak memiliki jenis add-on Amazon EKS yang diinstal pada cluster Anda. Untuk mengelola sendiri add-on, selesaikan langkah-langkah yang tersisa dalam prosedur ini untuk memperbarui add-on. Jika nomor versi dikembalikan, Anda memiliki jenis add-on Amazon EKS yang diinstal pada cluster Anda. Untuk memperbaruinya, gunakan

prosedur di [the section called “Perbarui add-on”](#), daripada menggunakan prosedur ini. Jika Anda tidak terbiasa dengan perbedaan antara jenis add-on, lihat [the section called “Add-on Amazon EKS”](#).

2. Lihat versi gambar kontainer mana yang saat ini diinstal di cluster Anda.

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |  
cut -d : -f 3
```

Contoh output adalah sebagai berikut.

```
v1.19.6-eksbuild.1
```

Output Anda mungkin tidak menyertakan nomor build.

3. Backup pengaturan Anda saat ini sehingga Anda dapat mengonfigurasi pengaturan yang sama setelah Anda memperbarui versi Anda.

```
kubectl get daemonset aws-node -n kube-system -o yaml > aws-k8s-cni-old.yaml
```

Untuk meninjau versi yang tersedia dan membiasakan diri dengan perubahan dalam versi yang ingin Anda perbarui, lihat [rilis](#) di GitHub. Perhatikan bahwa kami sarankan memperbarui ke hal yang sama `major.minor.patch` versi yang tercantum dalam tabel versi terbaru yang tersedia, bahkan jika versi yang lebih baru tersedia di GitHub. Untuk tabel versi terbaru yang tersedia, lihat [the section called “Versi Amazon VPC CNI”](#). Versi build yang tercantum dalam tabel tidak ditentukan dalam versi yang dikelola sendiri yang tercantum. GitHub Perbarui versi Anda dengan menyelesaikan tugas di salah satu opsi berikut:

- Jika Anda tidak memiliki pengaturan khusus untuk add-on, jalankan perintah di bawah `To apply this release:` judul GitHub untuk [rilis](#) yang Anda perbarui.
- Jika Anda memiliki pengaturan khusus, unduh file manifes dengan perintah berikut. Ubah <https://raw.githubusercontent.com/aws/amazon-vpc-cni-k8s/v1.19.6/config/master/aws-k8s-cni.yaml> ke URL untuk rilis GitHub yang Anda perbarui.

```
curl -O https://raw.githubusercontent.com/aws/amazon-vpc-cni-k8s/v1.19.6/config/  
master/aws-k8s-cni.yaml
```

Jika perlu, ubah manifes dengan pengaturan kustom dari cadangan yang Anda buat pada langkah sebelumnya, lalu terapkan manifes yang dimodifikasi ke kluster Anda. Jika node Anda

tidak memiliki akses ke repositori Amazon EKS Amazon ECR pribadi tempat gambar ditarik (lihat baris yang dimulai dengan `image: manifes`), maka Anda harus mengunduh gambar, menyalinnya ke repositori Anda sendiri, dan memodifikasi manifes untuk menarik gambar dari repositori Anda. Untuk informasi selengkapnya, lihat [the section called “Salin gambar ke repositori”](#).

```
kubectl apply -f aws-k8s-cni.yaml
```

4. Konfirmasikan bahwa versi baru sekarang diinstal pada cluster Anda.

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |  
cut -d : -f 3
```

Contoh output adalah sebagai berikut.

```
v1.19.6
```

5. (Opsional) Instal `cni-metrics-helper` ke cluster Anda. Ini mengikis elastic network interface dan informasi alamat IP, menggabungkannya pada tingkat cluster, dan menerbitkan metrik ke Amazon CloudWatch. Untuk informasi lebih lanjut, lihat [cni-metrics-helper](#) di GitHub.

Konfigurasi plugin Amazon VPC CNI untuk menggunakan IRSA

Plugin [Amazon VPC CNI untuk Kubernetes](#) adalah plugin jaringan untuk jaringan Pod di cluster Amazon EKS. Plugin ini bertanggung jawab untuk mengalokasikan alamat IP VPC ke pod Kubernetes dan mengonfigurasi jaringan yang diperlukan untuk Pod pada setiap node.

Note

Plugin Amazon VPC CNI juga mendukung Amazon EKS Pod Identities. Untuk informasi selengkapnya, lihat [the section called “Tetapkan peran IAM”](#).

Plugin:

- Memerlukan AWS izin Identity and Access Management (IAM). Jika klaster Anda menggunakan IPv4 keluarga, izin akan ditentukan dalam kebijakan [AmazonEKS_CNI_Policy](#) AWS terkelola. Jika klaster Anda menggunakan IPv6 keluarga, maka izin harus ditambahkan ke kebijakan IAM

yang Anda buat; untuk instruksi, lihat [the section called “Buat kebijakan IAM untuk cluster yang menggunakan keluarga IPv6”](#). Anda dapat melampirkan kebijakan ke peran IAM node Amazon EKS, atau ke peran IAM terpisah. Untuk petunjuk untuk melampirkan kebijakan ke peran IAM node Amazon EKS, lihat [the section called “IAM role simpul”](#). Kami menyarankan Anda menetapkannya ke peran terpisah, seperti yang dijelaskan dalam topik ini.

- Membuat dan dikonfigurasi untuk menggunakan akun layanan Kubernetes yang diberi nama `aws-node` saat di-deploy. Akun layanan terikat ke `clusterrole` Kubernetes bernama `aws-node`, yang ditugaskan izin Kubernetes yang diperlukan.

Note

Pod untuk plugin Amazon VPC CNI untuk Kubernetes memiliki akses ke izin yang ditetapkan ke [peran IAM node Amazon EKS](#), kecuali jika Anda memblokir akses ke IMDS. Untuk informasi selengkapnya, lihat [Membatasi akses ke profil instance yang ditetapkan ke node pekerja](#).

- Membutuhkan cluster Amazon EKS yang ada. Untuk menyebarkan satu, lihat [Memulai](#).
- Memerlukan penyedia OpenID Connect (OIDC) AWS Identity and Access Management (IAM) yang sudah ada untuk klaster Anda. Untuk menentukan apakah Anda sudah memiliki satu, atau harus membuat satu, lihat [the section called “Penyedia IAM OIDC”](#).

Langkah 1: Buat plugin Amazon VPC CNI untuk peran IAM Kubernetes

1. Tentukan keluarga IP cluster Anda.

```
aws eks describe-cluster --name my-cluster | grep ipFamily
```

Contoh output adalah sebagai berikut.

```
"ipFamily": "ipv4"
```

Output dapat kembali `ipv6` sebagai gantinya.

2. Buat peran IAM. Anda dapat menggunakan `eksctl` atau `kubect1` dan AWS CLI untuk membuat peran IAM Anda.

eksctl

- Buat peran IAM dan lampirkan kebijakan IAM ke peran dengan perintah yang cocok dengan keluarga IP cluster Anda. Perintah membuat dan menerapkan AWS CloudFormation tumpukan yang membuat peran IAM, melampirkan kebijakan yang Anda tentukan padanya, dan membubuhi keterangan akun layanan aws-node Kubernetes yang ada dengan ARN dari peran IAM yang dibuat.
- IPv4

Ganti *my-cluster* dengan nilai milik Anda sendiri.

```
eksctl create iamserviceaccount \  
  --name aws-node \  
  --namespace kube-system \  
  --cluster my-cluster \  
  --role-name AmazonEKSVPCCNIRole \  
  --attach-policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \  
  --override-existing-serviceaccounts \  
  --approve
```

- IPv6

Ganti *my-cluster* dengan nilai milik Anda sendiri. Ganti *111122223333* dengan ID akun Anda dan ganti *AmazonEKS_CNI_IPv6_Policy* dengan nama IPv6 polis Anda. Jika Anda tidak memiliki IPv6 kebijakan, lihat [the section called “Buat kebijakan IAM untuk cluster yang menggunakan keluarga IPv6”](#) untuk membuatnya. Untuk digunakan IPv6 dengan cluster Anda, itu harus memenuhi beberapa persyaratan. Untuk informasi selengkapnya, lihat [the section called “IPv6”](#).

```
eksctl create iamserviceaccount \  
  --name aws-node \  
  --namespace kube-system \  
  --cluster my-cluster \  
  --role-name AmazonEKSVPCCNIRole \  
  --attach-policy-arn arn:aws:iam::111122223333:policy/  
AmazonEKS_CNI_IPv6_Policy \  
  --override-existing-serviceaccounts \  
  --approve
```

kubectl dan CLI AWS

- i. Lihat URL penyedia OIDC klaster Anda.

```
aws eks describe-cluster --name my-cluster --query
"cluster.identity.oidc.issuer" --output text
```

Contoh output adalah sebagai berikut.

```
https://oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE
```

Jika tidak ada output yang dikembalikan, maka Anda harus [membuat penyedia IAM OIDC untuk](#) cluster Anda.

- ii. Salin isi berikut ke file bernama *vpc-cni-trust-policy.json*. Ganti **111122223333** dengan ID akun Anda dan **EXAMPLED539D4633E53DE1B71EXAMPLE** dengan output yang dikembalikan pada langkah sebelumnya. Ganti *region-code* dengan AWS Wilayah tempat kluster Anda berada.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:oidc-provider/
oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "oidc.eks.region-code.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com",
          "oidc.eks.region-code.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-system:aws-
node"
        }
      }
    }
  ]
}
```

- iii. Buat peran. Anda dapat mengganti *AmazonEKSVPCCNIRole* dengan nama apa pun yang Anda pilih.

```
aws iam create-role \
  --role-name AmazonEKSVPCCNIRole \
  --assume-role-policy-document file://"vpc-cni-trust-policy.json"
```

iv. Lampirkan kebijakan IAM yang diperlukan ke peran tersebut. Jalankan perintah yang cocok dengan keluarga IP cluster Anda.

- IPv4

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name AmazonEKSVPCCNIRole
```

- IPv6

Ganti **111122223333** dengan ID akun Anda dan **AmazonEKS_CNI_IPv6_Policy** dengan nama IPv6 polis Anda. Jika Anda tidak memiliki IPv6 kebijakan, lihat [the section called “Buat kebijakan IAM untuk cluster yang menggunakan keluarga IPv6”](#) untuk membuatnya. Untuk digunakan IPv6 dengan cluster Anda, itu harus memenuhi beberapa persyaratan. Untuk informasi selengkapnya, lihat [the section called “IPv6”](#).

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::111122223333:policy/AmazonEKS_CNI_IPv6_Policy \
  --role-name AmazonEKSVPCCNIRole
```

v. Jalankan perintah berikut untuk membuat anotasi akun aws-node layanan dengan ARN dari peran IAM yang Anda buat sebelumnya. Ganti **example values** dengan nilai-nilai milik Anda sendiri.

```
kubectl annotate serviceaccount \
  -n kube-system aws-node \
  eks.amazonaws.com/role-arn=arn:aws:iam::111122223333:role/
AmazonEKSVPCCNIRole
```

3. (Opsional) Konfigurasi tipe endpoint AWS Security Token Service yang digunakan oleh akun layanan Kubernetes Anda. Untuk informasi selengkapnya, lihat [the section called “Titik akhir STS”](#).

Langkah 2: Menerapkan ulang plugin Amazon VPC CNI untuk Kubernetes Pods

1. Hapus dan buat ulang Pod yang ada yang terkait dengan akun layanan untuk menerapkan variabel lingkungan kredensi. Anotasi tidak diterapkan pada Pod yang saat ini berjalan tanpa anotasi. Perintah berikut menghapus aws-node DaemonSet Pod yang ada dan menerapkannya dengan anotasi akun layanan.

```
kubectl delete Pods -n kube-system -l k8s-app=aws-node
```

2. Konfirmasikan bahwa semua Pod dimulai ulang.

```
kubectl get pods -n kube-system -l k8s-app=aws-node
```

3. Jelaskan salah satu Pod dan verifikasi bahwa variabel `AWS_WEB_IDENTITY_TOKEN_FILE` dan `AWS_ROLE_ARN` lingkungan ada. Ganti `cpjw7` dengan nama salah satu Pod Anda yang dikembalikan pada output dari langkah sebelumnya.

```
kubectl describe pod -n kube-system aws-node-cpjw7 | grep 'AWS_ROLE_ARN:\  
AWS_WEB_IDENTITY_TOKEN_FILE:'
```

Contoh output adalah sebagai berikut.

```
AWS_ROLE_ARN: arn:aws: iam::111122223333:role/AmazonEKSVPCCNIRole  
  AWS_WEB_IDENTITY_TOKEN_FILE: /var/run/secrets/eks.amazonaws.com/  
serviceaccount/token  
  AWS_ROLE_ARN: arn:aws: iam::111122223333:role/AmazonEKSVPCCNIRole  
  AWS_WEB_IDENTITY_TOKEN_FILE: /var/run/secrets/eks.amazonaws.com/  
serviceaccount/token
```

Dua set hasil duplikat dikembalikan karena Pod berisi dua kontainer. Kedua kontainer memiliki nilai yang sama.

Jika Pod Anda menggunakan endpoint AWS Regional, maka baris berikut juga dikembalikan pada output sebelumnya.

```
AWS_STS_REGIONAL_ENDPOINTS=regional
```

Langkah 3: Hapus kebijakan CNI dari peran IAM node

Jika [peran IAM node Amazon EKS](#) Anda saat ini memiliki [IPv6 kebijakan AmazonEKS_CNI_Policy](#) IAM (IPv4) yang dilampirkan padanya, dan Anda telah membuat peran IAM terpisah, melampirkan kebijakan ke dalamnya, dan menetapkan ke akun layanan aws-node Kubernetes, maka sebaiknya Anda menghapus kebijakan tersebut dari peran node Anda dengan perintah CLI yang cocok dengan keluarga IP AWS kluster Anda. Ganti *AmazonEKSNodeRole* dengan nama peran node Anda.

- IPv4

```
aws iam detach-role-policy --role-name AmazonEKSNodeRole --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
```

- IPv6

Ganti *111122223333* dengan ID akun Anda dan *AmazonEKS_CNI_IPv6_Policy* dengan nama IPv6 polis Anda.

```
aws iam detach-role-policy --role-name AmazonEKSNodeRole --policy-arn arn:aws:iam::111122223333:policy/AmazonEKS_CNI_IPv6_Policy
```

Buat kebijakan IAM untuk cluster yang menggunakan keluarga **IPv6**

Jika Anda membuat kluster yang menggunakan IPv6 keluarga dan kluster memiliki versi 1.10.1 atau yang lebih baru dari plugin Amazon VPC CNI untuk add-on Kubernetes yang dikonfigurasi, maka Anda perlu membuat kebijakan IAM yang dapat Anda tetapkan ke peran IAM. Jika Anda memiliki cluster yang sudah ada yang tidak Anda konfigurasikan dengan IPv6 keluarga saat Anda membuatnya, maka untuk menggunakannya IPv6, Anda harus membuat cluster baru. Untuk informasi selengkapnya tentang penggunaan IPv6 dengan kluster Anda, lihat [the section called "IPv6"](#).

1. Salin teks berikut dan simpan ke file bernama `vpc-cni-ipv6-policy.json`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```

        "Action": [
            "ec2:AssignIpv6Addresses",
            "ec2:DescribeInstances",
            "ec2:DescribeTags",
            "ec2:DescribeNetworkInterfaces",
            "ec2:DescribeInstanceTypes"
        ],
        "Resource": "*"
    },
    {
        "Effect": "Allow",
        "Action": [
            "ec2:CreateTags"
        ],
        "Resource": [
            "arn:aws:ec2:*:*:network-interface/*"
        ]
    }
]
}

```

2. Buat kebijakan IAM.

```
aws iam create-policy --policy-name AmazonEKS_CNI_IPv6_Policy --policy-document
file://vpc-cni-ipv6-policy.json
```

Pelajari tentang mode dan konfigurasi VPC CNI

Plugin Amazon VPC CNI untuk Kubernetes menyediakan jaringan untuk Pod. Gunakan tabel berikut untuk mempelajari lebih lanjut tentang fitur jaringan yang tersedia.

Fitur jaringan	Pelajari selengkapnya
Konfigurasi kluster Anda untuk menetapkan IPv6 alamat ke cluster, Pod, dan layanan	the section called “IPv6”
Menggunakan Terjemahan Alamat Jaringan IPv4 Sumber untuk Pod	the section called “Lalu lintas keluar”
Membatasi lalu lintas jaringan ke dan dari Pod	the section called “Batasi lalu lintas”

Fitur jaringan	Pelajari selengkapnya
Sesuaikan antarmuka jaringan sekunder di node	the section called “Jaringan kustom”
Tingkatkan alamat IP untuk node Anda	the section called “Meningkatkan alamat IP”
Menggunakan grup keamanan untuk lalu lintas jaringan Pod	the section called “Grup keamanan untuk Pod”
Menggunakan beberapa antarmuka jaringan untuk Pod	the section called “Beberapa antarmuka”

Pelajari tentang IPv6 alamat ke kluster, Pod, dan layanan

Berlaku untuk: Pod dengan EC2 instans Amazon dan Pod Fargate

Secara default, Kubernetes memberikan IPv4 alamat ke Pod dan layanan Anda. Alih-alih menetapkan IPv4 alamat ke Pod dan layanan Anda, Anda dapat mengonfigurasi kluster untuk menetapkan IPv6 alamat ke Pod dan layanan Anda. Amazon EKS tidak mendukung Pod atau layanan bertumpuk ganda, meskipun Kubernetes melakukannya. Akibatnya, Anda tidak dapat menetapkan keduanya IPv4 dan IPv6 alamat ke Pod dan layanan Anda.

Anda memilih keluarga IP mana yang ingin Anda gunakan untuk cluster Anda saat Anda membuatnya. Anda tidak dapat mengubah keluarga setelah membuat cluster.

Untuk tutorial menerapkan IPv6 kluster Amazon EKS, lihat [the section called “Deploy”](#).

Berikut ini adalah pertimbangan untuk menggunakan fitur ini:

IPv6 Dukungan fitur

- Tidak ada dukungan Windows: Pod dan layanan Windows tidak didukung.
- Diperlukan EC2 node berbasis Nitro: Anda hanya dapat menggunakan IPv6 dengan node AWS Amazon atau EC2 Fargate berbasis Nitro.
- EC2 dan node Fargate didukung: Anda dapat menggunakan IPv6 dengan dengan EC2 node [the section called “Grup keamanan untuk Pod”](#) Amazon dan node Fargate.
- Outposts tidak didukung: Anda tidak dapat menggunakan IPv6 dengan. [Amazon EKS di AWS Outposts](#)

- FSx untuk Lustre tidak didukung: Tidak [the section called “Amazon FSx untuk Lustre”](#) didukung.
- Jaringan khusus tidak didukung: Jika sebelumnya Anda digunakan [the section called “Jaringan kustom”](#) untuk membantu mengurangi kelelahan alamat IP, Anda dapat menggunakannya sebagai gantinya. IPv6 Anda tidak dapat menggunakan jaringan khusus dengan IPv6. Jika Anda menggunakan jaringan khusus untuk isolasi jaringan, maka Anda mungkin perlu terus menggunakan jaringan khusus dan IPv4 keluarga untuk cluster Anda.

Penugasan alamat IP

- Layanan Kubernetes: Layanan Kubernetes hanya diberi alamat. IPv6 Mereka tidak diberi IPv4 alamat.
- Pod: Pod diberi IPv6 alamat dan alamat host-lokal IPv4 . IPv4 Alamat host-lokal ditetapkan dengan menggunakan plugin CNI host-lokal yang dirantai dengan VPC CNI dan alamatnya tidak dilaporkan ke bidang kontrol Kubernetes. Ini hanya digunakan ketika pod perlu berkomunikasi dengan IPv4 sumber daya eksternal di VPC Amazon lain atau internet. IPv4 Alamat host-lokal mendapat SNATed (oleh VPC CNI) ke IPv4 alamat utama ENI primer dari node pekerja.
- Pod dan layanan: Pod dan layanan hanya menerima IPv6 alamat, bukan IPv4 alamat. Ketika Pod perlu berkomunikasi dengan IPv4 endpoint eksternal, mereka menggunakan NAT pada node itu sendiri. Kemampuan NAT bawaan ini menghilangkan kebutuhan untuk [DNS64 dan NAT64](#). Untuk lalu lintas yang membutuhkan akses internet publik, lalu lintas Pod adalah alamat jaringan sumber yang diterjemahkan ke alamat IP publik.
- Alamat perutean: Ketika sebuah Pod berkomunikasi di luar VPC, IPv6 alamat aslinya dipertahankan (tidak diterjemahkan ke alamat node). IPv6 Lalu lintas ini diarahkan langsung melalui gateway internet atau gateway internet khusus egress.
- Node: Semua node diberi IPv6 alamat IPv4 dan.
- Pod Fargate: Setiap Pod Fargate menerima IPv6 alamat dari CIDR yang ditentukan untuk subnet yang digunakan. Unit perangkat keras dasar yang menjalankan Fargate Pods mendapatkan IPv6 alamat unik IPv4 dan dari CIDRs yang ditetapkan ke subnet tempat unit perangkat keras digunakan.

Cara menggunakan **IPv6** EKS

- Buat cluster baru: Anda harus membuat cluster baru dan menentukan bahwa Anda ingin menggunakan IPv6 keluarga untuk cluster itu. Anda tidak dapat mengaktifkan IPv6 keluarga

untuk klaster yang Anda perbarui dari versi sebelumnya. Untuk petunjuk tentang cara membuat klaster baru, lihat [Pertimbangan](#).

- Gunakan VPC CNI terbaru: Terapkan versi Amazon VPC CNI atau yang lebih baru. 1.10.1 Versi ini atau yang lebih baru digunakan secara default. Setelah menerapkan add-on, Anda tidak dapat menurunkan versi add-on Amazon VPC CNI ke versi yang lebih rendah daripada 1.10.1 tanpa terlebih dahulu menghapus semua node di semua grup node di cluster Anda.
- Konfigurasi CNI VPC untuk **IPv6**: Jika Anda menggunakan EC2 node Amazon, Anda harus mengonfigurasi add-on Amazon VPC CNI dengan delegasi awalan IP dan IPv6. Jika Anda memilih IPv6 keluarga saat membuat cluster Anda, 1.10.1 versi add-on default ke konfigurasi ini. Ini adalah kasus untuk add-on Amazon EKS yang dikelola sendiri atau Amazon. Untuk informasi selengkapnya tentang delegasi awalan IP, lihat [the section called “Meningkatkan alamat IP”](#)
- Konfigurasi **IPv4** dan **IPv6** alamat: Saat Anda membuat cluster, VPC dan subnet yang Anda tentukan harus memiliki blok IPv6 CIDR yang ditetapkan ke VPC dan subnet yang Anda tentukan. Mereka juga harus memiliki blok IPv4 CIDR yang ditugaskan untuk mereka. Ini karena, meskipun Anda hanya ingin menggunakan IPv6, VPC masih memerlukan blok IPv4 CIDR untuk berfungsi. Untuk informasi selengkapnya, lihat [Mengaitkan blok IPv6 CIDR dengan VPC Anda](#) di Panduan Pengguna Amazon VPC.
- Tetapkan IPv6 alamat otomatis ke node: Saat Anda membuat node, Anda harus menentukan subnet yang dikonfigurasi untuk menetapkan alamat secara otomatis. IPv6 Jika tidak, Anda tidak dapat menyebarkan node Anda. Secara default, konfigurasi ini dinonaktifkan. Untuk informasi selengkapnya, lihat [Memodifikasi atribut IPv6 pengalamatan untuk subnet Anda](#) di Panduan Pengguna Amazon VPC.
- Atur tabel rute yang akan digunakan **IPv6**: Tabel rute yang ditetapkan ke subnet Anda harus memiliki rute untuk IPv6 alamat. Untuk informasi selengkapnya, lihat [Memigrasi ke IPv6](#) dalam Panduan Pengguna Amazon VPC.
- Tetapkan grup keamanan untuk **IPv6**: Grup keamanan Anda harus mengizinkan IPv6 alamat. Untuk informasi selengkapnya, lihat [Memigrasi ke IPv6](#) dalam Panduan Pengguna Amazon VPC.
- Mengatur load balancer: Gunakan versi 2.3.1 atau yang lebih baru dari AWS Load Balancer Controller untuk memuat keseimbangan aplikasi HTTP menggunakan [the section called “Penyeimbangan beban aplikasi”](#) [the section called “Menyeimbangkan beban jaringan”](#) lalu lintas jaringan atau IPv6 ke Pod dengan penyeimbang beban dalam mode IP, tetapi bukan mode instance. Untuk informasi selengkapnya, lihat [the section called “AWS Pengontrol Load Balancer”](#).
- Tambahkan kebijakan **IPv6** IAM: Anda harus melampirkan kebijakan IPv6 IAM ke node IAM atau peran IAM CNI Anda. Di antara keduanya, kami sarankan Anda melampirkannya ke peran IAM CNI. Untuk informasi selengkapnya, lihat [the section called “Buat kebijakan IAM untuk cluster yang](#)

[menggunakan keluarga IPv6](#) dan [the section called “Langkah 1: Buat plugin Amazon VPC CNI untuk peran IAM Kubernetes”](#).

- Evaluasi semua komponen: Lakukan evaluasi menyeluruh terhadap aplikasi, add-on Amazon EKS, dan AWS layanan yang Anda integrasikan sebelum menerapkan clusterIPv6. Ini untuk memastikan bahwa semuanya berfungsi seperti yang diharapkanIPv6.
- Tambahkan grup node yang **BootstrapArguments** dikelola sendiri: Saat membuat grup simpul yang dikelola sendiri dalam kluster yang menggunakan IPv6 keluarga, data pengguna harus menyertakan yang berikut BootstrapArguments untuk file [bootstrap.sh](#) yang berjalan saat node start up. Ganti *your-cidr* dengan rentang IPv6 CIDR dari VPC cluster Anda.

```
--ip-family ipv6 --service-ipv6-cidr your-cidr
```

Jika Anda tidak tahu IPv6 CIDR rentang untuk cluster Anda, Anda dapat melihatnya dengan perintah berikut (memerlukan versi AWS CLI 2.4.9 atau yang lebih baru).

```
aws eks describe-cluster --name my-cluster --query  
cluster.kubernetesNetworkConfig.serviceIpv6Cidr --output text
```

Menyebarkan **IPv6** kluster Amazon EKS dan mengelola node Amazon Linux

Dalam tutorial ini, Anda menerapkan VPC IPv6 Amazon, kluster Amazon EKS bersama IPv6 keluarga, dan grup node terkelola dengan node Amazon EC2 Amazon Linux. Anda tidak dapat menyebarkan node Amazon EC2 Windows dalam sebuah IPv6 cluster. Anda juga dapat menerapkan node Fargate ke cluster Anda, meskipun instruksi tersebut tidak disediakan dalam topik ini untuk kesederhanaan.

Prasyarat

Selesaikan yang berikut ini sebelum Anda memulai tutorial:

Instal dan konfigurasikan alat dan sumber daya berikut yang Anda perlukan untuk membuat dan mengelola kluster Amazon EKS.

- Kami menyarankan Anda membiasakan diri dengan semua pengaturan dan menyebarkan cluster dengan pengaturan yang memenuhi kebutuhan Anda. Untuk informasi lebih lanjut, lihat [the section called “Membuat kluster”](#), [the section called “Grup simpul terkelola”](#), dan [Pertimbangan](#) untuk topik ini. Anda hanya dapat mengaktifkan beberapa pengaturan saat membuat cluster Anda.

- Alat baris `kubectl` perintah diinstal pada perangkat Anda atau AWS CloudShell. Versinya bisa sama dengan atau hingga satu versi minor lebih awal atau lebih lambat dari versi Kubernetes dari kluster Anda. Misalnya, jika versi cluster Anda `1.29`, Anda dapat menggunakan `kubectl` versi `1.28`, `1.29`, atau `1.30` dengan itu. Untuk menginstal atau memutakhirkan `kubectl`, lihat [the section called “Siapkan kubectl dan eksctl”](#).
- Prinsip keamanan IAM yang Anda gunakan harus memiliki izin untuk bekerja dengan peran Amazon EKS IAM, peran terkait layanan, VPC, AWS CloudFormation dan sumber daya terkait. Untuk informasi selengkapnya, lihat [Tindakan](#) dan [Menggunakan peran terkait layanan](#) di Panduan Pengguna IAM.
- Jika Anda menggunakan `eksctl`, instal versi `0.210.0` atau yang lebih baru di komputer Anda. Untuk menginstal atau memperbaruinya, lihat [Instalasi](#) di `eksctl` dokumentasi.
- Versi `2.12.3` atau yang lebih baru atau versi `1.27.160` atau yang lebih baru dari AWS Command Line Interface (AWS CLI) diinstal dan dikonfigurasi pada perangkat Anda atau AWS CloudShell. Untuk memeriksa versi Anda saat ini, gunakan `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Package manager seperti `yum` atau `apt-get`, atau Homebrew untuk macOS seringkali merupakan beberapa versi di belakang versi terbaru CLI. AWS Untuk menginstal versi terbaru, lihat [Menginstal](#) dan [Konfigurasi cepat dengan aws configure](#) di Panduan Pengguna Antarmuka Baris AWS Perintah. Versi AWS CLI yang diinstal AWS CloudShell mungkin juga beberapa versi di belakang versi terbaru. Untuk memperbaruinya, lihat [Menginstal AWS CLI ke direktori home Anda](#) di AWS CloudShell Panduan Pengguna. Jika Anda menggunakan AWS CloudShell, Anda mungkin perlu [menginstal versi 2.12.3 atau yang lebih baru atau 1.27.160 atau yang lebih baru dari CLI AWS , karena versi AWS CLI default yang diinstal di mungkin versi sebelumnya](#). AWS CloudShell

Anda dapat menggunakan `eksctl` atau CLI untuk menyebarkan cluster. IPv6

Menyebarkan IPv6 cluster dengan `eksctl`

- a. Buat file `ipv6-cluster.yaml`. Salin perintah yang mengikuti ke perangkat Anda. Buat modifikasi berikut pada perintah sesuai kebutuhan dan kemudian jalankan perintah yang dimodifikasi:
 - Ganti `my-cluster` dengan nama untuk cluster Anda. Nama hanya dapat berisi karakter alfanumerik (peka huruf besar/kecil) dan tanda hubung. Itu harus dimulai dengan karakter alfanumerik dan tidak boleh lebih dari 100 karakter. Nama harus unik di dalam AWS Wilayah dan AWS akun tempat Anda membuat kluster.
 - Ganti `region-code` dengan AWS Wilayah apa pun yang didukung oleh Amazon EKS. Untuk daftar AWS Wilayah, lihat [titik akhir dan kuota Amazon EKS di panduan](#) Referensi AWS Umum.

- Nilai untuk `version` dengan versi cluster Anda. Untuk informasi selengkapnya, lihat [the section called “Versi Kubernetes”](#).
- Ganti `my-nodegroup` dengan nama untuk grup node Anda. Nama grup node tidak boleh lebih dari 63 karakter. Itu harus dimulai dengan huruf atau digit, tetapi juga dapat menyertakan tanda hubung dan garis bawah untuk karakter yang tersisa.
- Ganti `t3.medium` dengan [jenis instans Sistem AWS Nitro](#) apa pun.

```
cat >ipv6-cluster.yaml <<EOF
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code
  version: "X.XX"

kubernetesNetworkConfig:
  ipFamily: IPv6

addons:
  - name: vpc-cni
    version: latest
  - name: coredns
    version: latest
  - name: kube-proxy
    version: latest

iam:
  withOIDC: true

managedNodeGroups:
  - name: my-nodegroup
    instanceType: t3.medium
EOF
```

b. Buat cluster Anda.

```
eksctl create cluster -f ipv6-cluster.yaml
```

Pembuatan cluster membutuhkan waktu beberapa menit. Jangan lanjutkan sampai Anda melihat baris output terakhir, yang terlihat mirip dengan output berikut.

```
[...]
[#] EKS cluster "my-cluster" in "region-code" region is ready
```

c. Konfirmasikan bahwa Pod default adalah IPv6 alamat yang ditetapkan.

```
kubectl get pods -n kube-system -o wide
```

Contoh output adalah sebagai berikut.

NAME	READY	STATUS	RESTARTS	AGE	IP	NOMINATED NODE
READINESS GATES						
aws-node-rslts	1/1	Running	1	5m36s		
2600:1f13:b66:8200:11a5:ade0:c590:6ac8					ip-192-168-34-75.region-code.compute.internal	<none>
aws-node-t74jh	1/1	Running	0	5m32s		
2600:1f13:b66:8203:4516:2080:8ced:1ca9					ip-192-168-253-70.region-code.compute.internal	<none>
coredns-85d5b4454c-cw7w2	1/1	Running	0	56m		
2600:1f13:b66:8203:34e5::					ip-192-168-253-70.region-code.compute.internal	<none>
coredns-85d5b4454c-tx6n8	1/1	Running	0	56m		
2600:1f13:b66:8203:34e5::1					ip-192-168-253-70.region-code.compute.internal	<none>
kube-proxy-btpbk	1/1	Running	0	5m36s		
2600:1f13:b66:8200:11a5:ade0:c590:6ac8					ip-192-168-34-75.region-code.compute.internal	<none>
kube-proxy-jjk2g	1/1	Running	0	5m33s		
2600:1f13:b66:8203:4516:2080:8ced:1ca9					ip-192-168-253-70.region-code.compute.internal	<none>

d. Konfirmasikan bahwa layanan default adalah IPv6 alamat yang ditetapkan.

```
kubectl get services -n kube-system -o wide
```

Contoh output adalah sebagai berikut.

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE	
SELECTOR						
kube-dns	ClusterIP	fd30:3087:b6c2::a	<none>	53/UDP,53/TCP	57m	k8s-
app=kube-dns						

- e. (Optional) [Menerapkan aplikasi sampel](#) atau menerapkan [Load AWS Balancer](#) Controller dan aplikasi sampel untuk memuat aplikasi HTTP [the section called “Penyeimbangan beban aplikasi”](#) dengan atau lalu lintas [the section called “Menyeimbangkan beban jaringan”](#) jaringan dengan Pod. IPv6
- f. Setelah Anda selesai dengan cluster dan node yang Anda buat untuk tutorial ini, Anda harus membersihkan sumber daya yang Anda buat dengan perintah berikut.

```
eksctl delete cluster my-cluster
```

Menerapkan IPv6 cluster dengan CLI AWS

Important

- Anda harus menyelesaikan semua langkah dalam prosedur ini sebagai pengguna yang sama. Untuk memeriksa pengguna saat ini, jalankan perintah berikut:

```
aws sts get-caller-identity
```

- Anda harus menyelesaikan semua langkah dalam prosedur ini di shell yang sama. Beberapa langkah menggunakan variabel yang diatur dalam langkah sebelumnya. Langkah-langkah yang menggunakan variabel tidak akan berfungsi dengan baik jika nilai variabel diatur dalam shell yang berbeda. Jika Anda menggunakan [AWS CloudShell](#) untuk menyelesaikan prosedur berikut, ingatlah bahwa jika Anda tidak berinteraksi dengannya menggunakan keyboard atau pointer selama kurang lebih 20-30 menit, sesi shell Anda berakhir. Proses yang berjalan tidak dihitung sebagai interaksi.
- Instruksi ditulis untuk cangkang Bash, dan mungkin perlu disesuaikan dengan cangkang lainnya.

Ganti semua *example values* langkah prosedur ini dengan nilai Anda sendiri.

- a. Jalankan perintah berikut untuk mengatur beberapa variabel yang digunakan dalam langkah-langkah selanjutnya. Ganti *region-code* dengan AWS Wilayah tempat Anda ingin menyebarkan sumber daya Anda. Nilainya dapat berupa AWS Wilayah apa pun yang didukung oleh Amazon EKS. Untuk daftar AWS Wilayah, lihat [titik akhir dan kuota Amazon EKS di panduan Referensi AWS Umum](#). Ganti *my-cluster* dengan nama untuk cluster Anda. Nama hanya dapat berisi karakter alfanumerik (peka huruf besar/kecil) dan tanda hubung. Itu harus dimulai dengan karakter alfanumerik dan tidak boleh lebih dari 100 karakter. Nama harus unik di dalam AWS Wilayah dan AWS akun tempat Anda membuat klaster. Ganti *my-nodegroup* dengan nama untuk grup node Anda. Nama grup node tidak boleh lebih dari 63 karakter. Itu harus dimulai dengan huruf atau digit, tetapi juga dapat menyertakan tanda hubung dan garis bawah untuk karakter yang tersisa. Ganti *111122223333* dengan ID akun Anda.

```
export region_code=region-code
export cluster_name=my-cluster
export nodegroup_name=my-nodegroup
export account_id=111122223333
```

- b. Buat VPC Amazon dengan subnet publik dan pribadi yang memenuhi Amazon EKS dan persyaratan. IPv6

- i. Jalankan perintah berikut untuk mengatur variabel untuk nama AWS CloudFormation tumpukan Anda. Anda dapat mengganti *my-eks-ipv6-vpc* dengan nama apa pun yang Anda pilih.

```
export vpc_stack_name=my-eks-ipv6-vpc
```

- ii. Buat IPv6 VPC menggunakan template. AWS CloudFormation

```
aws cloudformation create-stack --region $region_code --stack-name $vpc_stack_name \
  --template-url https://s3.us-west-2.amazonaws.com/amazon-eks/
cloudformation/2020-10-29/amazon-eks-ipv6-vpc-public-private-subnets.yaml
```

Tumpukan membutuhkan waktu beberapa menit untuk membuatnya. Jalankan perintah berikut. Jangan melanjutkan ke langkah berikutnya sampai output dari perintah tersebut `CREATE_COMPLETE`.

```
aws cloudformation describe-stacks --region $region_code --stack-name
$vpc_stack_name --query Stacks[].StackStatus --output text
```

- iii. Ambil subnet publik yang dibuat. IDs

```
aws cloudformation describe-stacks --region $region_code --stack-name
  $vpc_stack_name \
    --query='Stacks[].Outputs[?OutputKey==`SubnetsPublic`].OutputValue' --output
  text
```

Contoh output adalah sebagai berikut.

```
subnet-0a1a56c486EXAMPLE, subnet-099e6ca77aEXAMPLE
```

iv. Aktifkan opsi IPv6 alamat tetapkan otomatis untuk subnet publik yang dibuat.

```
aws ec2 modify-subnet-attribute --region $region_code --subnet-id
  subnet-0a1a56c486EXAMPLE --assign-ipv6-address-on-creation
aws ec2 modify-subnet-attribute --region $region_code --subnet-id
  subnet-099e6ca77aEXAMPLE --assign-ipv6-address-on-creation
```

v. Ambil nama subnet dan grup keamanan yang dibuat oleh template dari AWS CloudFormation tumpukan yang digunakan dan simpan dalam variabel untuk digunakan di langkah selanjutnya.

```
security_groups=$(aws cloudformation describe-stacks --region $region_code --
stack-name $vpc_stack_name \
  --query='Stacks[].Outputs[?OutputKey==`SecurityGroups`].OutputValue' --output
  text)

public_subnets=$(aws cloudformation describe-stacks --region $region_code --stack-
name $vpc_stack_name \
  --query='Stacks[].Outputs[?OutputKey==`SubnetsPublic`].OutputValue' --output
  text)

private_subnets=$(aws cloudformation describe-stacks --region $region_code --
stack-name $vpc_stack_name \
  --query='Stacks[].Outputs[?OutputKey==`SubnetsPrivate`].OutputValue' --output
  text)

subnets=${public_subnets},${private_subnets}
```

c. Membuat IAM role klaster dan lampirkan kebijakan terkelola Amazon EKS IAM ke dalam peran itu. Cluster Kubernetes yang dikelola oleh Amazon EKS melakukan panggilan ke AWS layanan lain atas nama Anda untuk mengelola sumber daya yang Anda gunakan dengan layanan.

i. Jalankan perintah berikut untuk membuat `eks-cluster-role-trust-policy.json` file.

```
cat >eks-cluster-role-trust-policy.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

- ii. Jalankan perintah berikut untuk menetapkan variabel untuk nama peran Anda. Anda dapat mengganti *myAmazonEKSClusterRole* dengan nama apa pun yang Anda pilih.

```
export cluster_role_name=myAmazonEKSClusterRole
```

- iii. Buat peran.

```
aws iam create-role --role-name $cluster_role_name --assume-role-policy-document
file://"eks-cluster-role-trust-policy.json"
```

- iv. Ambil ARN dari peran IAM dan simpan dalam variabel untuk langkah selanjutnya.

```
CLUSTER_IAM_ROLE=$(aws iam get-role --role-name $cluster_role_name --
query="Role.Arn" --output text)
```

- v. Lampirkan kebijakan terkelola IAM Amazon EKS yang diperlukan untuk peran tersebut.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSClusterPolicy --role-name $cluster_role_name
```

- d. Buat cluster Anda.

```
aws eks create-cluster --region $region_code --name $cluster_name --kubernetes-
version 1.XX \
  --role-arn $CLUSTER_IAM_ROLE --resources-vpc-config subnetIds=
$subnets,securityGroupIds=$security_groups \
```

```
--kubernetes-network-config ipFamily=ipv6
```

- i. CATATAN: Anda mungkin menerima kesalahan bahwa salah satu Availability Zone dalam permintaan Anda tidak memiliki kapasitas yang cukup untuk membuat kluster Amazon EKS. Jika hal ini terjadi, output galat berisi Availability Zones yang dapat mendukung kluster baru. Cobalah untuk kembali membuat kluster dengan setidaknya dua subnet yang terletak di Availability Zones yang didukung untuk akun Anda. Untuk informasi selengkapnya, lihat [the section called “Kapasitas tidak mencukupi”](#).

Cluster membutuhkan beberapa menit untuk membuatnya. Jalankan perintah berikut. Jangan melanjutkan ke langkah berikutnya sampai output dari perintah ACTIVE.

```
aws eks describe-cluster --region $region_code --name $cluster_name --query
cluster.status
```

- e. Buat atau perbarui kubeconfig file untuk kluster Anda sehingga Anda dapat berkomunikasi dengan cluster Anda.

```
aws eks update-kubeconfig --region $region_code --name $cluster_name
```

Secara default, config file dibuat ~/.kube atau konfigurasi cluster baru ditambahkan ke config file yang ada di ~/.kube.

- f. Buat peran IAM node.

- i. Jalankan perintah berikut untuk membuat vpc-cni-ipv6-policy.json file.

```
cat >vpc-cni-ipv6-policy.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:AssignIpv6Addresses",
        "ec2:DescribeInstances",
        "ec2:DescribeTags",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeInstanceTypes"
      ],
      "Resource": "*"
    }
  ],
}
```

```

    {
      "Effect": "Allow",
      "Action": [
        "ec2:CreateTags"
      ],
      "Resource": [
        "arn:aws: ec2:*:*:network-interface/*"
      ]
    }
  ]
}
EOF

```

ii. Buat kebijakan IAM.

```
aws iam create-policy --policy-name AmazonEKS_CNI_IPv6_Policy --policy-document
file://vpc-cni-ipv6-policy.json
```

iii. Jalankan perintah berikut untuk membuat `node-role-trust-relationship.json` file.

```

cat >node-role-trust-relationship.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

```

iv. Jalankan perintah berikut untuk menetapkan variabel untuk nama peran Anda. Anda dapat mengganti *AmazonEKSNodeRole* dengan nama apa pun yang Anda pilih.

```
export node_role_name=AmazonEKSNodeRole
```

v. Buat peran IAM.


```
aws iam create-role --role-name $node_role_name --assume-role-policy-document
file://"node-role-trust-relationship.json"
```

vi. Lampirkan kebijakan IAM ke peran IAM.

```
aws iam attach-role-policy --policy-arn arn:aws: iam::$account_id:policy/
AmazonEKS_CNI_IPv6_Policy \
  --role-name $node_role_name
```

Important

Untuk kesederhanaan dalam tutorial ini, kebijakan dilampirkan ke peran IAM ini. Namun, dalam kluster produksi, kami merekomendasikan untuk melampirkan kebijakan ke peran IAM yang terpisah. Untuk informasi selengkapnya, lihat [the section called “Konfigurasi untuk IRSA”](#).

vii. Lampirkan dua kebijakan yang dikelola IAM yang diperlukan ke peran IAM.

```
aws iam attach-role-policy --policy-arn arn:aws: iam::aws:policy/
AmazonEKSWorkerNodePolicy \
  --role-name $node_role_name
aws iam attach-role-policy --policy-arn arn:aws: iam::aws:policy/
AmazonEC2ContainerRegistryReadOnly \
  --role-name $node_role_name
```

viii. Ambil ARN dari peran IAM dan simpan dalam variabel untuk langkah selanjutnya.

```
node_iam_role=$(aws iam get-role --role-name $node_role_name --query="Role.Arn" --
output text)
```

g. Buat grup node terkelola.

i. Lihat subnet yang Anda buat pada langkah sebelumnya. IDs

```
echo $subnets
```

Contoh output adalah sebagai berikut.

```
subnet-0a1a56c486EXAMPLE, subnet-099e6ca77aEXAMPLE, subnet-0377963d69EXAMPLE, subnet-0c05f819
```

- ii. Buat grup node. Ganti `0a1a56c486EXAMPLE099e6ca77aEXAMPLE`, `0377963d69EXAMPLE`, dan `0c05f819d5EXAMPLE` dengan nilai yang dikembalikan dalam output dari langkah sebelumnya. Pastikan untuk menghapus koma antara subnet IDs dari output sebelumnya dalam perintah berikut. Anda dapat mengganti `t3.medium` dengan [jenis instans Sistem AWS Nitro](#) apa pun.

```
aws eks create-nodegroup --region $region_code --cluster-name $cluster_name --
nodegroup-name $nodegroup_name \
  --subnets subnet-0a1a56c486EXAMPLE subnet-099e6ca77aEXAMPLE
subnet-0377963d69EXAMPLE subnet-0c05f819d5EXAMPLE \
  --instance-types t3.medium --node-role $node_iam_role
```

Grup node membutuhkan waktu beberapa menit untuk membuatnya. Jalankan perintah berikut. Jangan lanjutkan ke langkah berikutnya sampai output yang dikembalikan `ACTIVE`.

```
aws eks describe-nodegroup --region $region_code --cluster-name $cluster_name --
nodegroup-name $nodegroup_name \
  --query nodegroup.status --output text
```

- h. Konfirmasikan bahwa Pod default adalah IPv6 alamat yang ditetapkan di IP kolom.

```
kubectl get pods -n kube-system -o wide
```

Contoh output adalah sebagai berikut.

NAME	READY	STATUS	RESTARTS	AGE	IP	
	NODE					NOMINATED NODE
READINESS GATES						
aws-node-rslts	1/1	Running	1	5m36s		
2600:1f13:b66:8200:11a5:ade0:c590:6ac8					ip-192-168-34-75.region-	
code.compute.internal	<none>		<none>			
aws-node-t74jh	1/1	Running	0	5m32s		
2600:1f13:b66:8203:4516:2080:8ced:1ca9					ip-192-168-253-70.region-	
code.compute.internal	<none>		<none>			
coredns-85d5b4454c-cw7w2	1/1	Running	0	56m		
2600:1f13:b66:8203:34e5::					ip-192-168-253-70.region-	
code.compute.internal	<none>		<none>			
coredns-85d5b4454c-tx6n8	1/1	Running	0	56m		
2600:1f13:b66:8203:34e5::1					ip-192-168-253-70.region-	
code.compute.internal	<none>		<none>			

```

kube-proxy-btpbk          1/1      Running    0          5m36s
2600:1f13:b66:8200:11a5:ade0:c590:6ac8  ip-192-168-34-75.region-
code.compute.internal    <none>      <none>
kube-proxy-jjk2g          1/1      Running    0          5m33s
2600:1f13:b66:8203:4516:2080:8ced:1ca9  ip-192-168-253-70.region-
code.compute.internal    <none>      <none>

```

- i. Konfirmasikan bahwa layanan default diberikan IPv6 alamat di IP kolom.

```
kubectl get services -n kube-system -o wide
```

Contoh output adalah sebagai berikut.

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
SELECTOR					
kube-dns	ClusterIP	fd30:3087:b6c2::a	<none>	53/UDP,53/TCP	57m
app=kube-dns					k8s-

- j. (Opsional) [Menerapkan aplikasi sampel](#) atau menerapkan [Load AWS Balancer](#) Controller dan aplikasi sampel untuk memuat aplikasi HTTP [the section called “Penyeimbangan beban aplikasi”](#) dengan atau lalu lintas [the section called “Menyeimbangkan beban jaringan”](#) jaringan dengan Pod. IPv6
- k. Setelah Anda selesai dengan cluster dan node yang Anda buat untuk tutorial ini, Anda harus membersihkan sumber daya yang Anda buat dengan perintah berikut. Pastikan Anda tidak menggunakan sumber daya apa pun di luar tutorial ini sebelum menghapusnya.
- i. Jika Anda menyelesaikan langkah ini di shell yang berbeda dari saat Anda menyelesaikan langkah sebelumnya, tetapkan nilai semua variabel yang digunakan pada langkah sebelumnya, ganti *example values* dengan nilai yang Anda tentukan saat Anda menyelesaikan langkah sebelumnya. Jika Anda menyelesaikan langkah ini di shell yang sama dengan tempat Anda menyelesaikan langkah sebelumnya, lewati ke langkah berikutnya.

```

export region_code=region-code
export vpc_stack_name=my-eks-ipv6-vpc
export cluster_name=my-cluster
export nodegroup_name=my-nodegroup
export account_id=111122223333
export node_role_name=AmazonEKSNodeRole
export cluster_role_name=myAmazonEKSClusterRole

```

- ii. Hapus grup node Anda.

```
aws eks delete-nodegroup --region $region_code --cluster-name $cluster_name --  
nodegroup-name $nodegroup_name
```

Penghapusan membutuhkan waktu beberapa menit. Jalankan perintah berikut. Jangan melanjutkan ke langkah berikutnya jika ada output yang dikembalikan.

```
aws eks list-nodegroups --region $region_code --cluster-name $cluster_name --query  
nodegroups --output text
```

iii. Hapus klaster .

```
aws eks delete-cluster --region $region_code --name $cluster_name
```

Cluster membutuhkan beberapa menit untuk menghapus. Sebelum melanjutkan pastikan bahwa cluster dihapus dengan perintah berikut.

```
aws eks describe-cluster --region $region_code --name $cluster_name
```

Jangan lanjutkan ke langkah berikutnya sampai output Anda mirip dengan output berikut.

```
An error occurred (ResourceNotFoundException) when calling the DescribeCluster  
operation: No cluster found for name: my-cluster.
```

iv. Hapus sumber daya IAM yang Anda buat. Ganti *AmazonEKS_CNI_IPv6_Policy* dengan nama yang Anda pilih, jika Anda memilih nama yang berbeda dari yang digunakan pada langkah sebelumnya.

```
aws iam detach-role-policy --role-name $cluster_role_name --policy-arn arn:aws:  
iam::aws:policy/AmazonEKSClusterPolicy  
aws iam detach-role-policy --role-name $node_role_name --policy-arn arn:aws:  
iam::aws:policy/AmazonEKSWorkerNodePolicy  
aws iam detach-role-policy --role-name $node_role_name --policy-arn arn:aws:  
iam::aws:policy/AmazonEC2ContainerRegistryReadOnly  
aws iam detach-role-policy --role-name $node_role_name --policy-arn arn:aws: iam::  
$account_id:policy/AmazonEKS_CNI_IPv6_Policy  
aws iam delete-policy --policy-arn arn:aws: iam::$account_id:policy/  
AmazonEKS_CNI_IPv6_Policy  
aws iam delete-role --role-name $cluster_role_name  
aws iam delete-role --role-name $node_role_name
```

v. Hapus AWS CloudFormation tumpukan yang membuat VPC.

```
aws cloudformation delete-stack --region $region_code --stack-name $vpc_stack_name
```

Aktifkan akses internet outbound untuk Pod

Berlaku untuk: Node IPv4 Fargate Linux, node Linux dengan instance Amazon EC2

Jika Anda menerapkan klaster Anda menggunakan IPv6 keluarga, maka informasi dalam topik ini tidak berlaku untuk klaster Anda, karena IPv6 alamat tidak diterjemahkan jaringan. Untuk informasi selengkapnya tentang penggunaan IPv6 dengan klaster Anda, lihat [the section called “IPv6”](#).

Secara default, setiap Pod di klaster Anda diberi IPv4 alamat [pribadi](#) dari blok classless inter-domain routing (CIDR) yang terkait dengan VPC tempat Pod digunakan. Pod dalam VPC yang sama berkomunikasi satu sama lain menggunakan alamat IP pribadi ini sebagai titik akhir. [Ketika sebuah Pod berkomunikasi ke IPv4 alamat apa pun yang tidak berada dalam blok CIDR yang terkait dengan VPC Anda, plugin Amazon VPC CNI \(untuk Linux atau Windows\) menerjemahkan alamat Pod ke IPv4 alamat pribadi utama dari elastic network interface primer dari node tempat IPv4 Pod berjalan, secara default*.](#)

Note

Untuk node Windows, ada detail tambahan yang perlu dipertimbangkan. Secara default, [plugin VPC CNI untuk Windows](#) didefinisikan dengan konfigurasi jaringan di mana lalu lintas ke tujuan dalam VPC yang sama dikecualikan untuk SNAT. Ini berarti bahwa komunikasi VPC internal telah dinonaktifkan SNAT dan alamat IP yang dialokasikan ke Pod dapat dirutekan di dalam VPC. Tetapi lalu lintas ke tujuan di luar VPC memiliki sumber IP Pod yang diambil ke alamat IP utama ENI instance. Konfigurasi default untuk Windows ini memastikan bahwa pod dapat mengakses jaringan di luar VPC Anda dengan cara yang sama seperti instance host.

Karena perilaku ini:

- Pod Anda dapat berkomunikasi dengan sumber daya internet hanya jika node yang mereka jalankan memiliki alamat IP [publik](#) atau [elastis](#) yang ditetapkan untuk itu dan berada dalam [subnet publik](#). [Tabel rute](#) terkait subnet publik memiliki rute ke gateway internet. Kami merekomendasikan untuk menyebarkan node ke subnet pribadi, bila memungkinkan.

- Untuk versi plugin sebelumnya 1.8.0, sumber daya yang ada di jaringan atau VPCs yang terhubung ke VPC klaster Anda menggunakan [peering VPC](#), [VPC transit](#), atau Direct [AWS Connect](#) tidak dapat memulai komunikasi ke Pod Anda di belakang antarmuka jaringan elastis sekunder. Pod Anda dapat memulai komunikasi ke sumber daya tersebut dan menerima tanggapan darinya.

Jika salah satu dari pernyataan berikut benar di lingkungan Anda, maka ubah konfigurasi default dengan perintah berikut.

- Anda memiliki sumber daya di jaringan atau VPCs yang terhubung ke VPC klaster Anda menggunakan VPC [peering](#), [VPC transit](#), atau Direct [AWS Connect](#) yang perlu memulai komunikasi dengan Pod Anda menggunakan IPv4 alamat dan versi plugin Anda lebih awal dari itu. 1.8.0
- Pod Anda berada dalam [subnet pribadi](#) dan perlu berkomunikasi keluar ke internet. Subnet memiliki rute ke gateway [NAT](#).

```
kubectl set env daemonset -n kube-system aws-node AWS_VPC_K8S_CNI_EXTERNALSNAT=true
```

Note

Variabel konfigurasi `AWS_VPC_K8S_CNI_EXTERNALSNAT` dan `AWS_VPC_K8S_CNI_EXCLUDE_SNAT_CIDRS` CNI tidak berlaku untuk node Windows. Menonaktifkan SNAT tidak didukung untuk Windows. Adapun untuk mengecualikan daftar IPv4 CIDRs dari SNAT, Anda dapat menentukan ini dengan menentukan `ExcludedSnatCIDRs` parameter dalam skrip bootstrap Windows. Untuk informasi selengkapnya tentang penggunaan parameter ini, lihat [the section called “Parameter konfigurasi skrip bootstrap”](#).

Jaringan host

* Jika spesifikasi Pod berisi `hostNetwork=true` (default `false`), maka alamat IP-nya tidak diterjemahkan ke alamat yang berbeda. Ini adalah kasus untuk plugin Amazon VPC CNI untuk Kubernetes Pods yang berjalan di klaster Anda, secara default. kube-proxy Untuk Pod ini, alamat IP sama dengan alamat IP utama node, sehingga alamat IP Pod tidak diterjemahkan. Untuk informasi

selengkapnya tentang `hostNetwork` pengaturan Pod, lihat [inti PodSpec v1 di referensi](#) API Kubernetes.

Batasi lalu lintas Pod dengan kebijakan jaringan Kubernetes

Secara default, tidak ada batasan di Kubernetes untuk alamat IP, port, atau koneksi antara Pod apa pun di klaster Anda atau antara Pod Anda dan sumber daya di jaringan lain mana pun. Anda dapat menggunakan kebijakan jaringan Kubernetes untuk membatasi lalu lintas jaringan ke dan dari Pod Anda. Untuk informasi selengkapnya, lihat [Kebijakan Jaringan](#) dalam dokumentasi Kubernetes.

Jika Anda memiliki versi 1.13 atau sebelumnya dari plugin Amazon VPC CNI untuk Kubernetes di klaster Anda, Anda perlu menerapkan solusi pihak ketiga untuk menerapkan kebijakan jaringan Kubernetes ke klaster Anda. Versi 1.14 atau versi plugin yang lebih baru dapat menerapkan kebijakan jaringan, jadi Anda tidak perlu menggunakan solusi pihak ketiga. Dalam topik ini, Anda mempelajari cara mengonfigurasi klaster Anda untuk menggunakan kebijakan jaringan Kubernetes di klaster Anda tanpa menggunakan add-on pihak ketiga.

Kebijakan jaringan di plugin Amazon VPC CNI untuk Kubernetes didukung dalam konfigurasi berikut.

- Versi 1.14 atau yang lebih baru dari plugin Amazon VPC CNI untuk Kubernetes di cluster Anda.
- Cluster dikonfigurasi untuk IPv4 atau IPv6 alamat.
- Anda dapat menggunakan kebijakan jaringan dengan [grup keamanan untuk Pod](#). Dengan kebijakan jaringan, Anda dapat mengontrol semua komunikasi dalam cluster. Dengan grup keamanan untuk Pod, Anda dapat mengontrol akses ke AWS layanan dari aplikasi di dalam Pod.
- Anda dapat menggunakan kebijakan jaringan dengan jaringan kustom dan delegasi awalan.

Pertimbangan

Arsitektur

- Saat menerapkan plugin Amazon VPC CNI untuk kebijakan jaringan Kubernetes ke klaster Anda dengan plugin Amazon VPC CNI untuk Kubernetes, Anda hanya dapat menerapkan kebijakan tersebut ke node Amazon Linux. EC2 Anda tidak dapat menerapkan kebijakan ke Fargate atau Windows node.
- Kebijakan jaringan hanya berlaku salah satu IPv4 atau IPv6 alamat, tetapi tidak keduanya. Dalam sebuah IPv4 cluster, VPC CNI memberikan IPv4 alamat ke pod dan menerapkan kebijakan. IPv4 Dalam sebuah IPv6 cluster, VPC CNI memberikan IPv6 alamat ke pod dan menerapkan

kebijakan. IPv6 Aturan kebijakan IPv4 jaringan apa pun yang diterapkan pada IPv6 klaster diabaikan. Aturan kebijakan IPv6 jaringan apa pun yang diterapkan pada IPv4 klaster diabaikan.

Kebijakan Jaringan

- Kebijakan Jaringan hanya diterapkan pada Pod yang merupakan bagian dari Deployment. Pod mandiri yang tidak memiliki `metadata.ownerReferences` satu set tidak dapat menerapkan kebijakan jaringan.
- Anda dapat menerapkan beberapa kebijakan jaringan ke Pod yang sama. Ketika dua atau lebih kebijakan yang memilih Pod yang sama dikonfigurasi, semua kebijakan diterapkan ke Pod.
- Jumlah maksimum kombinasi unik port untuk setiap protokol di masing-masing `ingress:` atau `egress:` pemilih dalam kebijakan jaringan adalah 24.
- Untuk salah satu layanan Kubernetes Anda, port layanan harus sama dengan port kontainer. Jika Anda menggunakan port bernama, gunakan nama yang sama dalam spesifikasi layanan juga.

Migrasi

- Jika klaster Anda saat ini menggunakan solusi pihak ketiga untuk mengelola kebijakan jaringan Kubernetes, Anda dapat menggunakan kebijakan yang sama dengan plugin Amazon VPC CNI untuk Kubernetes. Namun Anda harus menghapus solusi yang ada sehingga tidak mengelola kebijakan yang sama.

Instalasi

- Fitur kebijakan jaringan membuat dan memerlukan Definisi Sumber Daya `PolicyEndpoint` Kustom (CRD) yang disebut `policyendpoints.networking.k8s.aws`. `PolicyEndpoint` objek Sumber Daya Kustom dikelola oleh Amazon EKS. Anda tidak boleh memodifikasi atau menghapus sumber daya ini.
- Jika Anda menjalankan pod yang menggunakan kredensial IAM peran instance atau terhubung ke EC2 IMDS, berhati-hatilah untuk memeriksa kebijakan jaringan yang akan memblokir akses ke IMDS. EC2 Anda mungkin perlu menambahkan kebijakan jaringan untuk mengizinkan akses ke EC2 IMDS. Untuk informasi selengkapnya, lihat [Metadata instans dan data pengguna](#) di EC2 Panduan Pengguna Amazon.

Pod yang menggunakan peran IAM untuk akun layanan atau EKS Pod Identity tidak mengakses EC2 IMDS.

- Plugin Amazon VPC CNI untuk Kubernetes tidak menerapkan kebijakan jaringan ke antarmuka jaringan tambahan untuk setiap pod, hanya antarmuka utama untuk setiap pod (). `eth0` Ini mempengaruhi arsitektur berikut:
 - IPv6 pod dengan `ENABLE_V4_EGRESS` variabel disetel ke `true`. Variabel ini memungkinkan fitur IPv4 keluar untuk menghubungkan IPv6 pod ke IPv4 titik akhir seperti yang berada di luar cluster. Fitur IPv4 jalan keluar bekerja dengan membuat antarmuka jaringan tambahan dengan alamat IPv4 loopback lokal.
 - Saat menggunakan plugin jaringan berantai seperti Multus. Karena plugin ini menambahkan antarmuka jaringan ke setiap pod, kebijakan jaringan tidak diterapkan ke plugin jaringan yang dirantai.

Batasi lalu lintas jaringan Pod dengan kebijakan jaringan Kubernetes

Anda dapat menggunakan kebijakan jaringan Kubernetes untuk membatasi lalu lintas jaringan ke dan dari Pod Anda. Untuk informasi selengkapnya, lihat [Kebijakan Jaringan](#) dalam dokumentasi Kubernetes.

Anda harus mengkonfigurasi berikut ini untuk menggunakan fitur ini:

1. Siapkan penegakan kebijakan saat startup Pod. Anda melakukan ini dalam `aws-node` wadah VPC CNI. `DaemonSet`
2. Aktifkan parameter kebijakan jaringan untuk add-on.
3. Konfigurasi kluster Anda untuk menggunakan kebijakan jaringan Kubernetes

Sebelum Anda mulai, tinjau pertimbangannya. Untuk informasi selengkapnya, lihat [the section called “Pertimbangan”](#).

Prasyarat

Berikut ini adalah prasyarat untuk fitur ini:

Versi cluster minimum

Sebuah kluster Amazon EKS yang sudah ada. Untuk menyebarkan satu, lihat [Memulai](#). Cluster harus menjalankan salah satu versi Kubernetes dan versi platform yang tercantum dalam tabel berikut. Perhatikan bahwa versi Kubernetes dan platform yang lebih lambat dari yang terdaftar juga didukung. Anda dapat memeriksa versi Kubernetes Anda saat ini dengan mengganti `my-cluster` perintah berikut dengan nama cluster Anda dan kemudian menjalankan perintah yang dimodifikasi:

```
aws eks describe-cluster
    --name my-cluster --query cluster.version --output
    text
```

Versi Kubernetes	Versi platform
1.27.4	eks.5
1.26.7	eks.6

Versi VPC CNI minimum

Versi 1.14 atau yang lebih baru dari plugin Amazon VPC CNI untuk Kubernetes di cluster Anda. Anda dapat melihat versi mana yang saat ini Anda miliki dengan perintah berikut.

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |
cut -d : -f 3
```

Jika versi Anda lebih awal dari 1.14, lihat [the section called “Update \(EKS add-on\)”](#) untuk meningkatkan ke versi 1.14 atau yang lebih baru.

Versi kernel Linux minimum

Node Anda harus memiliki versi kernel Linux 5.10 atau yang lebih baru. Anda dapat memeriksa versi kernel Anda dengan `uname -r`. Jika Anda menggunakan versi terbaru Amazon EKS yang dioptimalkan Amazon Linux, Amazon EKS mengoptimalkan Amazon Linux yang dipercepat AMIs, dan Bottlerocket AMIs, mereka sudah memiliki versi kernel yang diperlukan.

Amazon EKS mengoptimalkan versi AMI Amazon Linux yang dipercepat v20231116 atau yang lebih baru memiliki versi kernel 5.10.

Langkah 1: Siapkan penegakan kebijakan saat startup Pod

Plugin Amazon VPC CNI untuk Kubernetes mengonfigurasi kebijakan jaringan untuk pod secara paralel dengan penyediaan pod. Sampai semua kebijakan dikonfigurasi untuk pod baru, container di pod baru akan dimulai dengan kebijakan allow default. Ini disebut mode standar. Kebijakan allow default berarti bahwa semua lalu lintas masuk dan keluar diizinkan ke dan dari pod baru. Misalnya, pod tidak akan memiliki aturan firewall yang diberlakukan (semua lalu lintas diizinkan) sampai pod baru diperbarui dengan kebijakan aktif.

Dengan `NETWORK_POLICY_ENFORCING_MODE` variabel disetel `kestrict`, pod yang menggunakan VPC CNI dimulai dengan kebijakan penolakan default, kemudian kebijakan dikonfigurasi. Ini disebut mode ketat. Dalam mode ketat, Anda harus memiliki kebijakan jaringan untuk setiap titik akhir yang perlu diakses pod di kluster Anda. Perhatikan bahwa persyaratan ini berlaku untuk pod CoreDNS. Kebijakan penolakan default tidak dikonfigurasi untuk pod dengan jaringan Host.

Anda dapat mengubah kebijakan jaringan default dengan menyetel variabel lingkungan `NETWORK_POLICY_ENFORCING_MODE` ke `strict` dalam `aws-node` wadah CNI VPC. DaemonSet

```
env:  
  - name: NETWORK_POLICY_ENFORCING_MODE  
    value: "strict"
```

Langkah 2: Aktifkan parameter kebijakan jaringan untuk add-on

Fitur kebijakan jaringan menggunakan port 8162 pada node untuk metrik secara default. Juga, fitur yang digunakan port 8163 untuk probe kesehatan. Jika Anda menjalankan aplikasi lain di node atau di dalam pod yang perlu menggunakan port ini, aplikasi gagal dijalankan. Dalam versi VPC CNI `v1.14.1` atau yang lebih baru, Anda dapat mengubah port ini.

Gunakan prosedur berikut untuk mengaktifkan parameter kebijakan jaringan untuk add-on.

AWS Management Console

1. Buka [konsol Amazon EKS](#).
2. Di panel navigasi kiri, pilih Cluster, lalu pilih nama cluster yang ingin Anda konfigurasikan untuk add-on Amazon VPC CNI.
3. Pilih tab Add-ons.
4. Pilih kotak di kanan atas kotak add-on dan kemudian pilih Edit.
5. Pada Amazon VPC CNI halaman Konfigurasi:
 - a. Pilih versi `v1.14.0-eksbuild.3` atau yang lebih baru dalam daftar Versi.
 - b. Perluas pengaturan konfigurasi opsional.
 - c. Masukkan kunci `"enableNetworkPolicy"`: dan nilai JSON `"true"` dalam nilai Konfigurasi. Teks yang dihasilkan harus berupa objek JSON yang valid. Jika kunci dan nilai ini adalah satu-satunya data dalam kotak teks, kelilingi kunci dan nilai dengan kurawal kurawal. `{ }`

Contoh berikut mengaktifkan fitur kebijakan jaringan dan metrik serta probe kesehatan disetel ke nomor port default:

```
{
  "enableNetworkPolicy": "true",
  "nodeAgent": {
    "healthProbeBindAddr": "8163",
    "metricsBindAddr": "8162"
  }
}
```

Helm

Jika Anda telah menginstal plugin Amazon VPC CNI untuk Kuberneteshelm, Anda dapat memperbarui konfigurasi untuk mengubah port.

1. Jalankan perintah berikut untuk mengubah port. Tetapkan nomor port dalam nilai untuk kunci `nodeAgent.metricsBindAddr` atau kunci `nodeAgent.healthProbeBindAddr`, masing-masing.

```
helm upgrade --set nodeAgent.metricsBindAddr=8162 --set
nodeAgent.healthProbeBindAddr=8163 aws-vpc-cni --namespace kube-system eks/aws-vpc-
cni
```

kubectl

1. Buka `aws-node` DaemonSet di editor Anda.

```
kubectl edit daemonset -n kube-system aws-node
```

2. Ganti nomor port dalam argumen perintah berikut di dalam `aws-network-policy-agent` wadah dalam `args`: manifes daemonset VPC CNI `aws-node`.

```
- args:
  - --metrics-bind-addr=:8162
  - --health-probe-bind-addr=:8163
```

Langkah 3: Pasang sistem file Berkeley Packet Filter (BPF) pada node Anda

Anda harus me-mount sistem file Berkeley Packet Filter (BPF) pada setiap node Anda.

Note

Jika klaster Anda versi 1.27 atau yang lebih baru, Anda dapat melewati langkah ini karena semua Amazon EKS mengoptimalkan Amazon Linux dan Bottlerocket AMIs untuk 1.27 atau yang lebih baru sudah memiliki fitur ini.

Untuk semua versi cluster lainnya, jika Anda memutakhirkan Amazon EKS yang dioptimalkan Amazon Linux ke versi v20230703 atau yang lebih baru atau Anda memutakhirkan AMI Bottlerocket ke v1.0.2 versi atau yang lebih baru, Anda dapat melewati langkah ini.

1. Pasang sistem file Berkeley Packet Filter (BPF) pada setiap node Anda.

```
sudo mount -t bpf bpffs /sys/fs/bpf
```

2. Kemudian, tambahkan perintah yang sama ke data pengguna Anda di template peluncuran untuk Grup EC2 Auto Scaling Amazon Anda.

Langkah 4: Konfigurasi klaster Anda untuk menggunakan kebijakan jaringan Kubernetes

Anda dapat mengatur ini untuk add-on Amazon EKS atau add-on yang dikelola sendiri.

Pengaya Amazon EKS

Dengan menggunakan AWS CLI, Anda dapat mengonfigurasi cluster untuk menggunakan kebijakan jaringan Kubernetes dengan menjalankan perintah berikut. Ganti `my-cluster` dengan nama cluster Anda dan peran IAM ARN dengan peran yang Anda gunakan.

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-version  
v1.14.0-eksbuild.3 \  
--service-account-role-arn arn:aws:iam::123456789012:role/AmazonEKSVPCCNIRole \  
--resolve-conflicts PRESERVE --configuration-values '{"enableNetworkPolicy":  
"true"}'
```

Untuk mengonfigurasinya menggunakan AWS Management Console, ikuti langkah-langkah di bawah ini:

1. Buka [konsol Amazon EKS](#).
2. Di panel navigasi kiri, pilih Cluster, lalu pilih nama cluster yang ingin Anda konfigurasi untuk add-on Amazon VPC CNI.

3. Pilih tab Add-ons.
4. Pilih kotak di kanan atas kotak add-on dan kemudian pilih Edit.
5. Pada Amazon VPC CNI halaman Konfigurasi:
 - a. Pilih versi `v1.14.0-eksbuild.3` atau yang lebih baru dalam daftar Versi.
 - b. Perluas pengaturan konfigurasi opsional.
 - c. Masukkan kunci `"enableNetworkPolicy"`: dan nilai JSON `"true"` dalam nilai Konfigurasi. Teks yang dihasilkan harus berupa objek JSON yang valid. Jika kunci dan nilai ini adalah satu-satunya data dalam kotak teks, kelilingi kunci dan nilai dengan kurawal kurawal. `{ }` Contoh berikut menunjukkan kebijakan jaringan diaktifkan:

```
{ "enableNetworkPolicy": "true" }
```

Screenshot berikut menunjukkan contoh skenario ini.



EKS > Clusters > > Add-on > vpc-cni > Edit add-on

Configure Amazon VPC CNI

Amazon VPC CNI

Info

Listed by 	Category networking	Status  Active

Version

Select the version for this add-on.

v1.17.1-eksbuild.1

Select IAM role

Select an IAM role to use with this add-on. To create a new role, follow the instructions in the [Amazon EKS User Guide](#).

Optional configuration settings

Add-on configuration schema

Refer to the JSON schema below. The configuration values entered in the code editor will be validated against this schema.

```
{
  "$ref": "#/definitions/VpcCni",
  "$schema": "http://json-schema.org/draft-06/schema#",
  "definitions": {
    "Affinity": {
      "type": [
        "object",
        "null"
      ]
    },
    "EniConfig": {
      "additionalProperties": false,

```

Configuration values

Info

Specify any additional JSON or YAML configurations that should be applied to the add-on.

1

{ "enableNetworkPolicy": "true" }

Add-on yang dikelola sendiri

Helm

Jika Anda telah menginstal plugin Amazon VPC CNI untuk Kuberneteshelm, Anda dapat memperbarui konfigurasi untuk mengaktifkan kebijakan jaringan.

1. Jalankan perintah berikut untuk mengaktifkan kebijakan jaringan.

```
helm upgrade --set enableNetworkPolicy=true aws-vpc-cni --namespace kube-system eks/
aws-vpc-cni
```

kubectl

1. Buka amazon-vpc-cni ConfigMap di editor Anda.

```
kubectl edit configmap -n kube-system amazon-vpc-cni -o yaml
```

2. Tambahkan baris berikut ke data dalamConfigMap.

```
enable-network-policy-controller: "true"
```

Setelah Anda menambahkan baris, Anda ConfigMap akan terlihat seperti contoh berikut.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: amazon-vpc-cni
  namespace: kube-system
data:
  enable-network-policy-controller: "true"
```

3. Buka aws-node DaemonSet di editor Anda.

```
kubectl edit daemonset -n kube-system aws-node
```

- a. Ganti false dengan true dalam argumen perintah di --enable-network-policy=false dalam aws-network-policy-agent wadah args: dalam manifes daemonset VPC CNIaws-node.


```
- args:  
  - --enable-network-policy=true
```

Langkah 5. Langkah selanjutnya

Setelah Anda menyelesaikan konfigurasi, konfirmasikan bahwa `aws-node` pod berjalan di kluster Anda.

```
kubectl get pods -n kube-system | grep 'aws-node\|amazon'
```

Contoh output adalah sebagai berikut.

aws-node-gmqp7	2/2	Running	1 (24h ago)
24h			
aws-node-prnsh	2/2	Running	1 (24h ago)
24h			

Ada 2 kontainer di `aws-node` pod dalam versi 1.14 dan yang lebih baru. Di versi sebelumnya dan jika kebijakan jaringan dinonaktifkan, hanya ada satu kontainer di `aws-node` pod.

Anda sekarang dapat menerapkan kebijakan jaringan Kubernetes ke kluster Anda.

Untuk menerapkan kebijakan jaringan Kubernetes, Anda membuat `NetworkPolicy` objek Kubernetes dan menerapkannya ke kluster Anda. `NetworkPolicy` objek dicakup ke namespace. Anda menerapkan kebijakan untuk mengizinkan atau menolak lalu lintas antar Pod berdasarkan pemilih label, ruang nama, dan rentang alamat IP. Untuk informasi selengkapnya tentang membuat `NetworkPolicy` objek, lihat [Kebijakan Jaringan](#) dalam dokumentasi Kubernetes.

Penegakan `NetworkPolicy` objek Kubernetes diimplementasikan menggunakan Extended Berkeley Packet Filter (eBPF). Sehubungan dengan implementasi iptables berbasis, ia menawarkan karakteristik latensi dan kinerja yang lebih rendah, termasuk pengurangan pemanfaatan CPU dan menghindari pencarian berurutan. Selain itu, probe eBPF menyediakan akses ke data kaya konteks yang membantu men-debug masalah tingkat kernel yang kompleks dan meningkatkan observabilitas. Amazon EKS mendukung eksportir berbasis EBPF yang memanfaatkan probe untuk mencatat hasil kebijakan pada setiap node dan mengeksport data ke pengumpul log eksternal untuk membantu debugging. Untuk informasi selengkapnya, lihat dokumentasi [eBPF](#).

Nonaktifkan kebijakan jaringan Kubernetes untuk lalu lintas jaringan Amazon EKS Pod

Nonaktifkan kebijakan jaringan Kubernetes untuk berhenti membatasi lalu lintas jaringan Amazon EKS Pod

1. Daftar semua kebijakan jaringan Kubernetes.

```
kubectl get netpol -A
```

2. Hapus setiap kebijakan jaringan Kubernetes. Anda harus menghapus semua kebijakan jaringan sebelum menonaktifkan kebijakan jaringan.

```
kubectl delete netpol <policy-name>
```

3. Buka aws-node DaemonSet di editor Anda.

```
kubectl edit daemonset -n kube-system aws-node
```

4. Ganti true dengan false dalam argumen perintah di `--enable-network-policy=true` dalam `aws-network-policy-agent` wadah `args`: dalam manifes daemonset VPC CNIAws-node.

```
- args:
  - --enable-network-policy=false
```

Memecahkan masalah kebijakan jaringan Kubernetes Untuk Amazon EKS

Ini adalah panduan pemecahan masalah untuk fitur kebijakan jaringan Amazon VPC CNI.

Panduan ini mencakup:

- Instal informasi, izin CRD dan RBAC [the section called “policyendpointsCRD dan izin baru”](#)
- Log untuk memeriksa saat mendiagnosis masalah kebijakan jaringan [the section called “Log kebijakan jaringan”](#)
- Menjalankan koleksi alat eBPF SDK untuk memecahkan masalah
- Masalah dan solusi yang diketahui [the section called “Masalah dan solusi yang diketahui”](#)

Note

Perhatikan bahwa kebijakan jaringan hanya diterapkan pada pod yang dibuat oleh Deployment Kubernetes. Untuk batasan lebih lanjut dari kebijakan jaringan di VPC CNI, lihat [the section called “Pertimbangan”](#)

[Anda dapat memecahkan masalah dan menyelidiki koneksi jaringan yang menggunakan kebijakan jaringan dengan membaca log kebijakan Jaringan dan dengan menjalankan alat dari SDK eBPF.](#)

policyendpointsCRD dan izin baru

- CRD: `policyendpoints.networking.k8s.aws`
- Kubernetes API: dipanggil `apiservice v1.networking.k8s.io`
- Sumber daya Kubernetes: Kind: `NetworkPolicy`
- RBAC: `ClusterRole` disebut (`aws-nodeVPC CNI`), `ClusterRole` disebut `eks:network-policy-controller` (pengontrol kebijakan jaringan di bidang kontrol cluster EKS)

Untuk kebijakan jaringan, VPC CNI membuat baru CustomResourceDefinition (CRD) yang disebut `policyendpoints.networking.k8s.aws` VPC CNI harus memiliki izin untuk membuat CRD dan membuat CustomResources (CR) ini dan CRD lainnya yang diinstal oleh VPC CNI (`eniconfigs.crd.k8s.amazonaws.com`). Keduanya CRDs tersedia dalam [crds.yamlfile](#) di GitHub. Secara khusus, CNI VPC harus memiliki izin kata kerja “get”, “list”, dan “watch” untuk `policyendpoints`

Kebijakan Jaringan Kubernetes adalah bagian dari `apiservice panggilanv1.networking.k8s.io`, dan ini ada `apiversion: networking.k8s.io/v1` dalam file YAMM kebijakan Anda. VPC CNI DaemonSet harus memiliki izin untuk menggunakan bagian Kubernetes API ini.

Izin VPC CNI berada dalam panggilan. `ClusterRole aws-node` Perhatikan bahwa `ClusterRole` objek tidak dikelompokkan dalam ruang nama. Berikut ini `aws-node` menunjukkan cluster:

```
kubectl get clusterrole aws-node -o yaml
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
```

```
labels:
  app.kubernetes.io/instance: aws-vpc-cni
  app.kubernetes.io/managed-by: Helm
  app.kubernetes.io/name: aws-node
  app.kubernetes.io/version: v1.19.4
  helm.sh/chart: aws-vpc-cni-1.19.4
  k8s-app: aws-node
name: aws-node
rules:
- apiGroups:
  - crd.k8s.amazonaws.com
  resources:
  - eniconfigs
  verbs:
  - list
  - watch
  - get
- apiGroups:
  - ""
  resources:
  - namespaces
  verbs:
  - list
  - watch
  - get
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
  - list
  - watch
  - get
- apiGroups:
  - ""
  resources:
  - nodes
  verbs:
  - list
  - watch
  - get
- apiGroups:
  - ""
  - events.k8s.io
```

```

resources:
- events
verbs:
- create
- patch
- list
- apiGroups:
- networking.k8s.aws
resources:
- policyendpoints
verbs:
- get
- list
- watch
- apiGroups:
- networking.k8s.aws
resources:
- policyendpoints/status
verbs:
- get
- apiGroups:
- vpcresources.k8s.aws
resources:
- cninodes
verbs:
- get
- list
- watch
- patch

```

Juga, pengontrol baru berjalan di bidang kontrol setiap cluster EKS. Pengontrol menggunakan izin yang ClusterRole dipanggil `eks:network-policy-controller`. Berikut ini `eks:network-policy-controller` menunjukkan cluster:

```
kubectl get clusterrole eks:network-policy-controller -o yaml
```

```

apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    app.kubernetes.io/name: amazon-network-policy-controller-k8s
  name: eks:network-policy-controller

```

```
rules:
- apiGroups:
  - ""
  resources:
  - namespaces
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - ""
  resources:
  - services
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - networking.k8s.aws
  resources:
  - policyendpoints
  verbs:
  - create
  - delete
  - get
  - list
  - patch
  - update
  - watch
- apiGroups:
  - networking.k8s.aws
  resources:
  - policyendpoints/finalizers
  verbs:
  - update
- apiGroups:
```

```

- networking.k8s.aws
resources:
- policyendpoints/status
verbs:
- get
- patch
- update
- apiGroups:
- networking.k8s.io
resources:
- networkpolicies
verbs:
- get
- list
- patch
- update
- watch

```

Log kebijakan jaringan

Setiap keputusan oleh VPC CNi apakah koneksi diizinkan atau ditolak oleh kebijakan jaringan dicatat dalam log aliran. Log kebijakan jaringan pada setiap node menyertakan log aliran untuk setiap pod yang memiliki kebijakan jaringan. Log kebijakan jaringan disimpan di `/var/log/aws-routed-eni/network-policy-agent.log`. Contoh berikut adalah dari sebuah `network-policy-agent.log` file:

```

{"level":"info","timestamp":"2023-05-30T16:05:32.573Z","logger":"ebpf-
client","msg":"Flow Info: ","Src
IP":"192.168.87.155","Src Port":38971,"Dest IP":"64.6.160","Dest
Port":53,"Proto":"UDP","Verdict":"ACCEPT"}

```

Log kebijakan jaringan dinonaktifkan secara default. Untuk mengaktifkan log kebijakan jaringan, ikuti langkah-langkah berikut:

Note

Log kebijakan jaringan memerlukan 1 vCPU tambahan untuk `aws-network-policy-agent` wadah dalam manifes VPC CNi. `aws-node DaemonSet`

Pengaya Amazon EKS

AWS Management Console

- a. Buka [konsol Amazon EKS](#).
- b. Di panel navigasi kiri, pilih Cluster, lalu pilih nama cluster yang ingin Anda konfigurasi untuk add-on Amazon VPC CNI.
- c. Pilih tab Add-ons.
- d. Pilih kotak di kanan atas kotak add-on dan kemudian pilih Edit.
- e. Pada *Amazon VPC CNI* halaman Konfigurasi:
 - i. Pilih versi `v1.14.0-eksbuild.3` atau yang lebih baru dalam daftar dropdown Versi.
 - ii. Perluas pengaturan konfigurasi opsional.
 - iii. Masukkan kunci JSON tingkat atas `"nodeAgent"`: dan nilai adalah objek dengan kunci `"enablePolicyEventLogs"`: dan nilai `"true"` dalam nilai Konfigurasi. Teks yang dihasilkan harus berupa objek JSON yang valid. Contoh berikut menunjukkan kebijakan jaringan dan log kebijakan jaringan diaktifkan, dan log kebijakan jaringan dikirim ke CloudWatch Log:

```
{
  "enableNetworkPolicy": "true",
  "nodeAgent": {
    "enablePolicyEventLogs": "true"
  }
}
```

Screenshot berikut menunjukkan contoh skenario ini.



[EKS](#) > [Clusters](#) > > [Add-on](#) > [vpc-cni](#) > **Edit add-on**

Configure Amazon VPC CNI

Amazon VPC CNI [Info](#)

Listed by



Category

networking

Status

🟢 Active

Version

Select the version for this add-on.

v1.17.1-eksbuild.1

Select IAM role

Select an IAM role to use with this add-on. To create a new role, follow the instructions in the [Amazon EKS User Guide](#).

▼



▼ Optional configuration settings

Add-on configuration schema

Refer to the JSON schema below. The configuration values entered in the code editor will be validated against this schema.

```
{
  "$ref": "#/definitions/VpcCni",
  "$schema": "http://json-schema.org/draft-06/schema#",
  "definitions": {
    "Affinity": {
      "type": [
        "object",
        "null"
      ]
    },
  },
  "EniConfig": {
    "additionalProperties": false,
  }
}
```

Configuration values [Info](#)

Specify any additional JSON or YAML configurations that should be applied to the add-on.

```
1 {  
2   "enableNetworkPolicy": "true",  
3   "nodeAgent": {  
4     "enablePolicyEventLogs": "true"  
5   }  
6 }
```

AWS CLI

- a. Jalankan perintah AWS CLI berikut. Ganti `my-cluster` dengan nama cluster Anda dan ganti peran IAM ARN dengan peran yang Anda gunakan.

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-  
version v1.14.0-eksbuild.3 \  
    --service-account-role-arn arn:aws:iam::123456789012:role/AmazonEKSVPCCNIRole  
\  
    --resolve-conflicts PRESERVE --configuration-values '{"nodeAgent":  
{"enablePolicyEventLogs": "true"}}'
```

Add-on yang dikelola sendiri

Helm

Jika Anda telah menginstal plugin Amazon VPC CNI untuk Kuberneteshelm, Anda dapat memperbarui konfigurasi untuk menulis log kebijakan jaringan.

- a. Jalankan perintah berikut untuk mengaktifkan kebijakan jaringan.

```
helm upgrade --set nodeAgent.enablePolicyEventLogs=true aws-vpc-cni --namespace  
kube-system eks/aws-vpc-cni
```

kubectl

Jika Anda telah menginstal plugin Amazon VPC CNI untuk Kuberneteskubectl, Anda dapat memperbarui konfigurasi untuk menulis log kebijakan jaringan.

- a. Buka `aws-node DaemonSet` di editor Anda.

```
kubectl edit daemonset -n kube-system aws-node
```

- b. Ganti `false` dengan `true` dalam argumen perintah `--enable-policy-event-logs=false` di dalam `aws-network-policy-agent` wadah `args:` dalam manifes VPC `aws-node DaemonSet CNI`.

```
- args:  
  - --enable-policy-event-logs=true
```

Kirim log kebijakan jaringan ke Amazon CloudWatch Logs

Anda dapat memantau log kebijakan jaringan menggunakan layanan seperti Amazon CloudWatch Logs. Anda dapat menggunakan metode berikut untuk mengirim log kebijakan jaringan ke CloudWatch Log.

Untuk kluster EKS, log kebijakan akan ditempatkan di bawah `/aws/eks/cluster-name/cluster/` dan untuk kluster K8S yang dikelola sendiri, log akan ditempatkan di bawah `/aws/k8s-cluster/cluster/`

Kirim log kebijakan jaringan dengan plugin Amazon VPC CNI untuk Kubernetes

Jika Anda mengaktifkan kebijakan jaringan, kontainer kedua akan ditambahkan ke `aws-node` pod untuk agen node. Agen node ini dapat mengirim log kebijakan jaringan ke CloudWatch Log.

Note

Hanya log kebijakan jaringan yang dikirim oleh agen node. Log lain yang dibuat oleh VPC CNI tidak disertakan.

Prasyarat

- Tambahkan izin berikut sebagai bait atau kebijakan terpisah ke peran IAM yang Anda gunakan untuk CNI VPC.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups",
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "*"
    }
  ]
}
```

```
}
```

Pengaya Amazon EKS

AWS Management Console

- a. Buka [konsol Amazon EKS](#).
- b. Di panel navigasi kiri, pilih Cluster, lalu pilih nama cluster yang ingin Anda konfigurasi untuk add-on Amazon VPC CNI.
- c. Pilih tab Add-ons.
- d. Pilih kotak di kanan atas kotak add-on dan kemudian pilih Edit.
- e. Pada *Amazon VPC CNI* halaman Konfigurasi:
 - i. Pilih versi `v1.14.0-eksbuild.3` atau yang lebih baru dalam daftar dropdown Versi.
 - ii. Perluas pengaturan konfigurasi opsional.
 - iii. Masukkan kunci JSON tingkat atas `"nodeAgent"`: dan nilai adalah objek dengan kunci `"enableCloudWatchLogs"`: dan nilai `"true"` dalam nilai Konfigurasi. Teks yang dihasilkan harus berupa objek JSON yang valid. Contoh berikut menunjukkan kebijakan jaringan dan log kebijakan jaringan diaktifkan, dan log dikirim ke CloudWatch Log:

```
{
  "enableNetworkPolicy": "true",
  "nodeAgent": {
    "enablePolicyEventLogs": "true",
    "enableCloudWatchLogs": "true",
  }
}
```

Screenshot berikut menunjukkan contoh skenario ini.



[EKS](#) > [Clusters](#) > > [Add-on](#) > [vpc-cni](#) > **Edit add-on**

Configure Amazon VPC CNI

Amazon VPC CNI [Info](#)

Listed by



Category

networking

Status

🟢 Active

Version

Select the version for this add-on.

v1.17.1-eksbuild.1

Select IAM role

Select an IAM role to use with this add-on. To create a new role, follow the instructions in the [Amazon EKS User Guide](#).

Downloaded from <http://ajphaphysiol.org/> on September 11, 2015



▼ Optional configuration settings

Add-on configuration schema

Refer to the JSON schema below. The configuration values entered in the code editor will be validated against this schema.

```
{
  "$ref": "#/definitions/VpcCni",
  "$schema": "http://json-schema.org/draft-06/schema#",
  "definitions": {
    "Affinity": {
      "type": [
        "object",
        "null"
      ]
    },
  },
  "EniConfig": {
    "additionalProperties": false,
  }
}
```

Configuration values [Info](#)

Specify any additional JSON or YAML configurations that should be applied to the add-on.

```
1 {
2   "enableNetworkPolicy": "true",
3   "nodeAgent": {
4     "enablePolicyEventLogs": "true",
5     "enableCloudWatchLogs": "true"
6   }
7 }
```

AWS CLI

- a. Jalankan perintah AWS CLI berikut. Ganti `my-cluster` dengan nama cluster Anda dan ganti peran IAM ARN dengan peran yang Anda gunakan.

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-version v1.14.0-eksbuild.3 \
    --service-account-role-arn arn:aws:iam::123456789012:role/AmazonEKSVPCCNIRole \
    --resolve-conflicts PRESERVE --configuration-values '{"nodeAgent": {"enablePolicyEventLogs": "true", "enableCloudWatchLogs": "true"}}'
```

Add-on yang dikelola sendiri

Helm

Jika Anda telah menginstal plugin Amazon VPC CNI untuk Kuberneteshelm, Anda dapat memperbarui konfigurasi untuk mengirim log kebijakan jaringan ke Log. CloudWatch

- a. Jalankan perintah berikut untuk mengaktifkan log kebijakan jaringan dan mengirimkannya ke CloudWatch Log.

```
helm upgrade --set nodeAgent.enablePolicyEventLogs=true --set nodeAgent.enableCloudWatchLogs=true aws-vpc-cni --namespace kube-system eks/aws-vpc-cni
```

kubectl

- a. Buka `aws-node DaemonSet` di editor Anda.

```
kubectl edit daemonset -n kube-system aws-node
```

- b. Ganti `false` dengan `true` dalam dua argumen perintah `--enable-policy-event-logs=false` dan `--enable-cloudwatch-logs=false` di dalam `aws-network-policy-agent` wadah `args`: di manifes VPC `aws-node DaemonSet CNI`.

```
- args:
  - --enable-policy-event-logs=true
  - --enable-cloudwatch-logs=true
```

Kirim log kebijakan jaringan dengan Fluent Bit **DaemonSet**

Jika Anda menggunakan Fluent Bit dalam a DaemonSet untuk mengirim log dari node Anda, Anda dapat menambahkan konfigurasi untuk menyertakan log kebijakan jaringan dari kebijakan jaringan. Anda dapat menggunakan konfigurasi contoh berikut:

```
[INPUT]
  Name          tail
  Tag           eksnp.*
  Path          /var/log/aws-routed-eni/network-policy-agent*.log
  Parser        json
  DB            /var/log/aws-routed-eni/flb_npagent.db
  Mem_Buf_Limit 5MB
  Skip_Long_Lines On
  Refresh_Interval 10
```

Termasuk eBPF SDK

Plugin Amazon VPC CNI untuk Kubernetes menginstal koleksi alat eBPF SDK pada node. Anda dapat menggunakan alat SDK eBPF untuk mengidentifikasi masalah dengan kebijakan jaringan. Misalnya, perintah berikut mencantumkan program yang berjalan pada node.

```
sudo /opt/cni/bin/aws-eks-na-cli ebpf progs
```

Untuk menjalankan perintah ini, Anda dapat menggunakan metode apa pun untuk terhubung ke node.

Masalah dan solusi yang diketahui

Bagian berikut menjelaskan masalah yang diketahui dengan fitur kebijakan jaringan Amazon VPC CNI dan solusinya.

Log kebijakan jaringan yang dihasilkan meskipun enable-policy-event-logs disetel ke false

Masalah: EKS VPC CNI menghasilkan log kebijakan jaringan bahkan ketika enable-policy-event-logs pengaturan disetel ke. false

Solusi: enable-policy-event-logs Pengaturan hanya menonaktifkan log “keputusan” kebijakan, tetapi tidak akan menonaktifkan semua pencatatan agen Kebijakan Jaringan. Perilaku ini didokumentasikan dalam [aws-network-policy-agent README](#) pada GitHub. Untuk sepenuhnya menonaktifkan logging, Anda mungkin perlu menyesuaikan konfigurasi logging lainnya.

Masalah pembersihan peta kebijakan jaringan

Masalah: Masalah dengan jaringan `policyendpoint` masih ada dan tidak dibersihkan setelah pod dihapus.

Solusi: Masalah ini disebabkan oleh masalah dengan add-on VPC CNI versi 1.19.3-eksbuild.1. Perbarui ke versi yang lebih baru dari add-on VPC CNI untuk mengatasi masalah ini.

Kebijakan jaringan tidak diterapkan

Masalah: Fitur kebijakan jaringan diaktifkan di plugin Amazon VPC CNI, tetapi kebijakan jaringan tidak diterapkan dengan benar.

Jika Anda membuat kebijakan jaringan `kind: NetworkPolicy` dan tidak mempengaruhi pod, periksa apakah objek `policyendpoint` dibuat di namespace yang sama dengan pod. Jika tidak ada `policyendpoint` objek di ruang nama, pengontrol kebijakan jaringan (bagian dari kluster EKS) tidak dapat membuat aturan kebijakan jaringan untuk diterapkan oleh agen kebijakan jaringan (bagian dari CNI VPC).

Solusi: Solusinya adalah memperbaiki izin VPC CNI `ClusterRole` (`aws-node:`) dan pengontrol `ClusterRole` kebijakan jaringan (`:`) dan mengizinkan tindakan ini di alat penegakan kebijakan apa pun seperti Kyverno. `eks:network-policy-controller` Pastikan bahwa kebijakan Kyverno tidak menghalangi pembuatan objek. `policyendpoint` Lihat bagian sebelumnya untuk izin izin yang diperlukan di. [the section called “policyendpointsCRD dan izin baru”](#)

Pod tidak kembali ke status penolakan default setelah penghapusan kebijakan dalam mode ketat

Masalah: Saat kebijakan jaringan diaktifkan dalam mode ketat, pod dimulai dengan kebijakan penolakan default. Setelah kebijakan diterapkan, lalu lintas diizinkan ke titik akhir yang ditentukan. Namun, ketika kebijakan dihapus, pod tidak kembali ke status penolakan default dan sebaliknya pergi ke status izinkan default.

Solusi: Masalah ini telah diperbaiki dalam rilis VPC CNI 1.19.3, yang mencakup rilis agen kebijakan jaringan 1.2.0. Setelah perbaikan, dengan mode ketat diaktifkan, setelah kebijakan dihapus, pod akan kembali ke status penolakan default seperti yang diharapkan.

Grup Keamanan untuk latensi startup Pod

Masalah: Saat menggunakan fitur Grup Keamanan untuk Pod di EKS, ada peningkatan latensi startup pod.

Solusi: Latensi disebabkan oleh pembatasan laju pada pengontrol sumber daya dari pembatasan API pada `CreateNetworkInterface` API, yang digunakan oleh pengontrol sumber daya VPC untuk membuat cabang untuk pod. ENIs Periksa batas API akun Anda untuk operasi ini dan pertimbangkan untuk meminta peningkatan batas jika diperlukan.

FailedScheduling karena `vpc.amazonaws.com/pod-eni` tidak mencukupi

Masalah: Pod gagal menjadwalkan dengan kesalahan: `FailedScheduling 2m53s (x28 over 137m) default-scheduler 0/5 nodes are available: 5 Insufficient vpc.amazonaws.com/pod-eni. preemption: 0/5 nodes are available: 5 No preemption victims found for incoming pod.`

Solusi: Seperti masalah sebelumnya, menetapkan Grup Keamanan ke pod meningkatkan latensi penjadwalan pod dan dapat meningkat melampaui ambang batas CNi untuk waktu untuk menambahkan setiap ENI, menyebabkan kegagalan untuk memulai pod. Ini adalah perilaku yang diharapkan saat menggunakan Grup Keamanan untuk Pod. Pertimbangkan implikasi penjadwalan saat merancang arsitektur beban kerja Anda.

Masalah konektivitas IPAM dan kesalahan segmentasi

Masalah: Beberapa kesalahan terjadi termasuk masalah konektivitas IPAM, permintaan pembatasan, dan kesalahan segmentasi:

- `Checking for IPAM connectivity ...`
- `Throttling request took 1.047064274s`
- `Retrying waiting for IPAM-D`
- `panic: runtime error: invalid memory address or nil pointer dereference`

Solusi: Masalah ini terjadi jika Anda menginstal `systemd-udev` pada AL2 023, karena file ditulis ulang dengan kebijakan yang melanggar. Ini dapat terjadi ketika memperbarui ke yang berbeda `releasever` yang memiliki paket yang diperbarui atau memperbarui paket itu sendiri secara manual. Hindari menginstal atau memperbarui `systemd-udev` pada AL2 023 node.

Gagal menemukan kesalahan nama perangkat

Masalah: Pesan kesalahan:

```
{"level":"error","ts":"2025-02-05T20:27:18.669Z","caller":"ebpf/bpf_client.go:578","msg":"failed to find device by name eni9ea69618bf0: %!w(netlink.LinkNotFoundError={0xc000115310})"}
```

Solusi: Masalah ini telah diidentifikasi dan diperbaiki di versi terbaru agen kebijakan jaringan Amazon VPC CNI (v1.2.0). Perbarui ke versi terbaru dari VPC CNI untuk mengatasi masalah ini.

Kerentanan CVE dalam gambar Multus CNI

Masalah: Laporan EKS ImageScan CVE yang Ditingkatkan mengidentifikasi kerentanan dalam versi gambar Multus CNI v4.1.4-eksbuild.2_thick.

Solusi: Perbarui ke versi baru gambar Multus CNI dan gambar Network Policy Controller baru, yang tidak memiliki kerentanan. Pemindai dapat diperbarui untuk mengatasi kerentanan yang ditemukan di versi sebelumnya.

Info Alur MENOLAK putusan di log

Masalah: Log kebijakan jaringan menunjukkan putusan DENY:

```
{"level":"info","ts":"2024-11-25T13:34:24.808Z","logger":"ebpf-client","caller":"events/events.go:193","msg":"Flow Info: ","Src IP":"","Src Port":9096,"Dest IP":"","Dest Port":56830,"Proto":"TCP","Verdict":"DENY"}
```

Solusi: Masalah ini telah diselesaikan dalam versi baru dari Network Policy Controller. Perbarui ke versi platform EKS terbaru untuk menyelesaikan masalah logging.

Pod-to-pod masalah komunikasi setelah bermigrasi dari Calico

Masalah: Setelah memutakhirkan kluster EKS ke versi 1.30 dan beralih dari Calico ke Amazon VPC CNI untuk kebijakan jaringan, pod-to-pod komunikasi gagal saat kebijakan jaringan diterapkan. Komunikasi dipulihkan ketika kebijakan jaringan dihapus.

Solusi: Agen kebijakan jaringan di VPC CNI tidak dapat memiliki banyak port yang ditentukan seperti yang dilakukan Calico. Sebagai gantinya, gunakan rentang port dalam kebijakan jaringan. Jumlah maksimum kombinasi unik port untuk setiap protokol di masing-masing ingress: atau egress: pemilih dalam kebijakan jaringan adalah 24. Gunakan rentang port untuk mengurangi jumlah port unik dan hindari batasan ini.

Agen kebijakan jaringan tidak mendukung pod mandiri

Masalah: Kebijakan jaringan yang diterapkan pada pod mandiri mungkin memiliki perilaku yang tidak konsisten.

Solusi: Agen Kebijakan Jaringan saat ini hanya mendukung pod yang di-deploy sebagai bagian dari deployment/replicaset. Jika kebijakan jaringan diterapkan ke pod mandiri, mungkin ada beberapa

ketidakkonsistenan dalam perilaku tersebut. Ini didokumentasikan di bagian atas halaman ini, [di the section called “Pertimbangan”](#), dan di [aws-network-policy-agent GitHub edisi #327](#) di GitHub. Menerapkan pod sebagai bagian dari penerapan atau replika untuk perilaku kebijakan jaringan yang konsisten.

Bintang demo kebijakan jaringan untuk Amazon EKS

Demo ini membuat layanan front-end, back-end, dan klien di cluster Amazon EKS Anda. Demo ini juga menciptakan antarmuka pengguna grafis manajemen yang menunjukkan jalur masuk dan keluar yang tersedia di antara setiap layanan. Kami menyarankan Anda menyelesaikan demo di kluster tempat Anda tidak menjalankan beban kerja produksi.

Sebelum Anda membuat kebijakan jaringan apa pun, semua layanan dapat berkomunikasi dua arah. Setelah Anda menerapkan kebijakan jaringan, Anda dapat melihat bahwa klien hanya dapat berkomunikasi dengan layanan front-end, dan back-end hanya menerima lalu lintas dari front-end.

1. Menerapkan layanan antarmuka pengguna front-end, back-end, klien, dan manajemen:

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-policies/calico/stars_policy_demo/create_resources.files/namespace.yaml
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-policies/calico/stars_policy_demo/create_resources.files/management-ui.yaml
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-policies/calico/stars_policy_demo/create_resources.files/backend.yaml
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-policies/calico/stars_policy_demo/create_resources.files/frontend.yaml
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-policies/calico/stars_policy_demo/create_resources.files/client.yaml
```

2. Lihat semua Pod di kluster.

```
kubectl get pods -A
```

Contoh output adalah sebagai berikut.

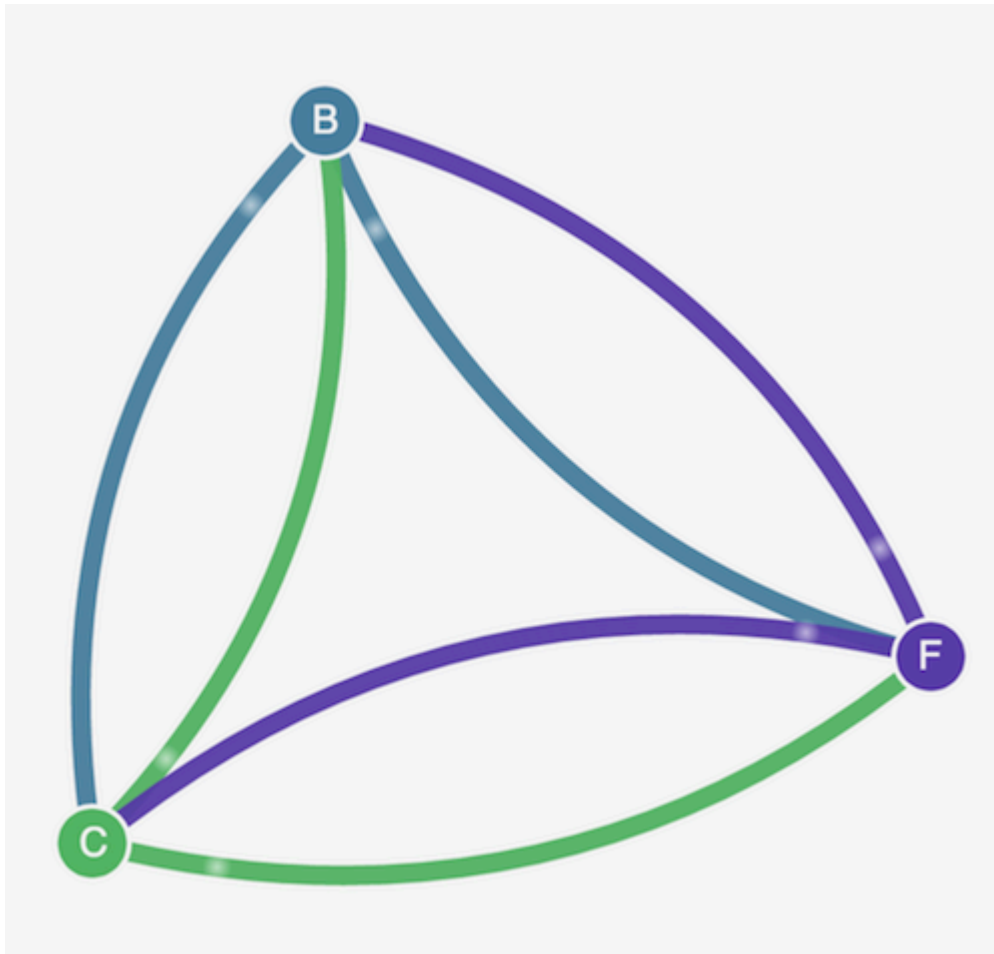
Dalam output Anda, Anda akan melihat pod di namespace yang ditunjukkan pada output berikut. Pod Anda dan jumlah pod di READY kolom berbeda dari yang ada di output berikut. **NAMES** Jangan lanjutkan sampai Anda melihat pod dengan nama yang mirip dan semuanya ada Running di STATUS kolom.

NAMESPACE	NAME	READY	STATUS
RESTARTS	AGE		
[...]			
client	client-xlffc	1/1	Running
5m19s			
[...]			
management-ui	management-ui-qrb2g	1/1	Running
5m24s			
stars	backend-sz87q	1/1	Running
5m23s			
stars	frontend-cscnf	1/1	Running
5m21s			
[...]			

- Untuk terhubung ke antarmuka pengguna manajemen, sambungkan ke layanan EXTERNAL-IP yang berjalan di kluster Anda:

```
kubectl get service/management-ui -n management-ui
```

- Buka browser ke lokasi dari langkah sebelumnya. Anda harus melihat antarmuka pengguna manajemen. Node C adalah layanan klien, simpul F adalah layanan front-end, dan simpul B adalah layanan back-end. Setiap node memiliki akses komunikasi penuh ke semua node lain, seperti yang ditunjukkan oleh garis tebal berwarna.



5. Terapkan kebijakan jaringan berikut di ruang nama `stars` dan `client` ruang nama untuk mengisolasi layanan satu sama lain:

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: default-deny
spec:
  podSelector:
    matchLabels: {}
```

Anda dapat menggunakan perintah berikut untuk menerapkan kebijakan ke kedua ruang nama:

```
kubectl apply -n stars -f https://raw.githubusercontent.com/aws-samples/eks-workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-policies/calico/stars_policy_demo/apply_network_policies.files/default-deny.yaml
```

```
kubectl apply -n client -f https://raw.githubusercontent.com/aws-samples/eks-workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-policies/calico/stars_policy_demo/apply_network_policies.files/default-deny.yaml
```

6. Segarkan peramban Anda. Anda melihat bahwa antarmuka pengguna manajemen tidak dapat lagi menjangkau salah satu node, sehingga tidak muncul di antarmuka pengguna.
7. Terapkan kebijakan jaringan yang berbeda berikut untuk memungkinkan antarmuka pengguna manajemen mengakses layanan. Terapkan kebijakan ini untuk mengizinkan UI:

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: stars
  name: allow-ui
spec:
  podSelector:
    matchLabels: {}
  ingress:
    - from:
      - namespaceSelector:
          matchLabels:
            role: management-ui
```

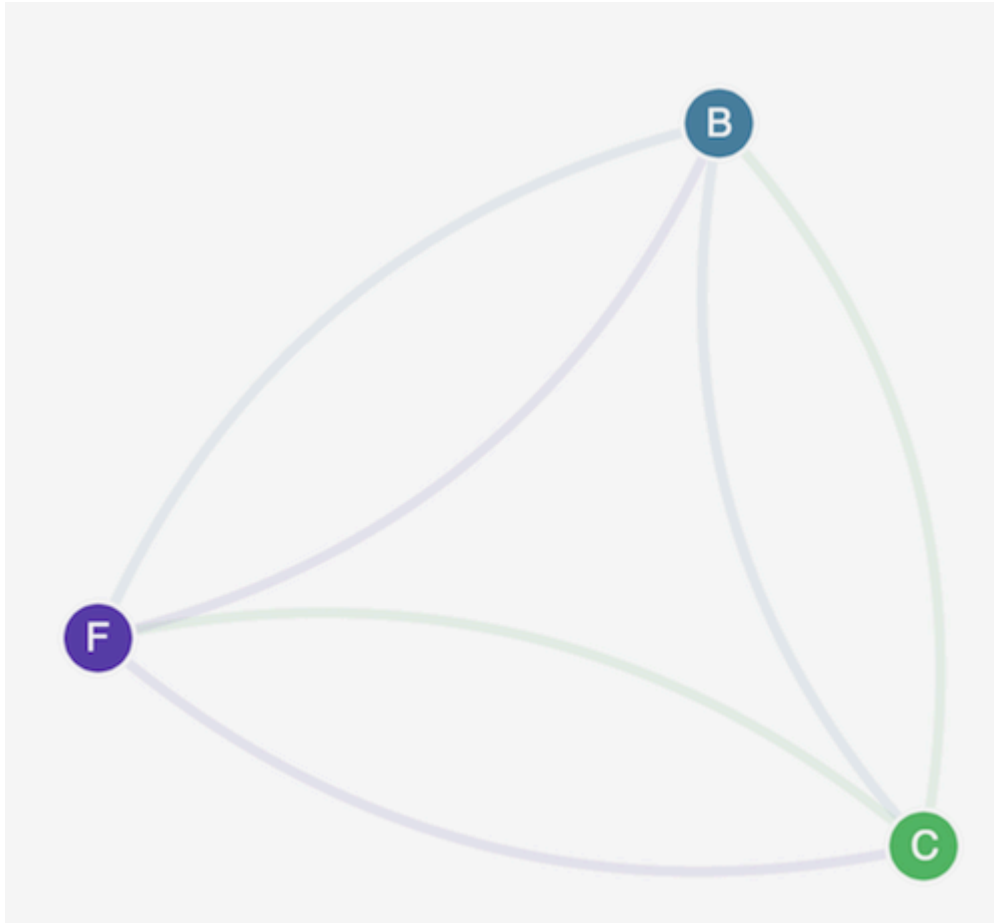
Terapkan kebijakan ini untuk mengizinkan klien:

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: client
  name: allow-ui
spec:
  podSelector:
    matchLabels: {}
  ingress:
    - from:
      - namespaceSelector:
          matchLabels:
            role: management-ui
```

Anda dapat menggunakan perintah berikut untuk menerapkan kedua kebijakan:

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-policies/calico/stars_policy_demo/apply_network_policies.files/allow-ui.yaml
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-policies/calico/stars_policy_demo/apply_network_policies.files/allow-ui-client.yaml
```

8. Segarkan peramban Anda. Anda melihat bahwa antarmuka pengguna manajemen dapat mencapai node lagi, tetapi node tidak dapat berkomunikasi satu sama lain.

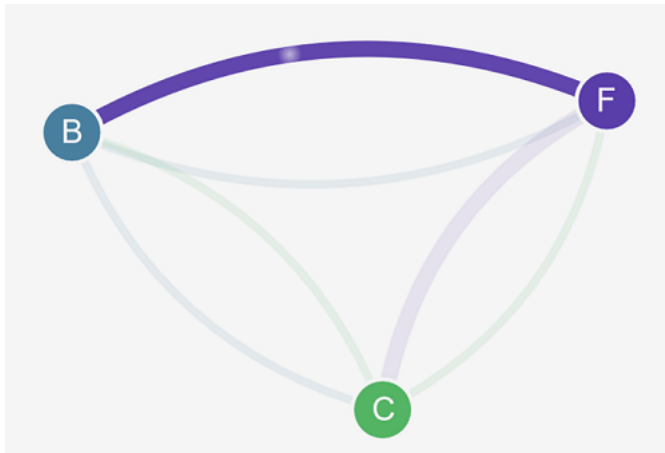


9. Terapkan kebijakan jaringan berikut untuk mengizinkan lalu lintas dari layanan front-end ke layanan back-end:

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: stars
  name: backend-policy
spec:
```

```
podSelector:
  matchLabels:
    role: backend
ingress:
  - from:
      - podSelector:
          matchLabels:
            role: frontend
    ports:
      - protocol: TCP
        port: 6379
```

10. Segarkan peramban Anda. Anda melihat bahwa front-end dapat berkomunikasi dengan back-end.



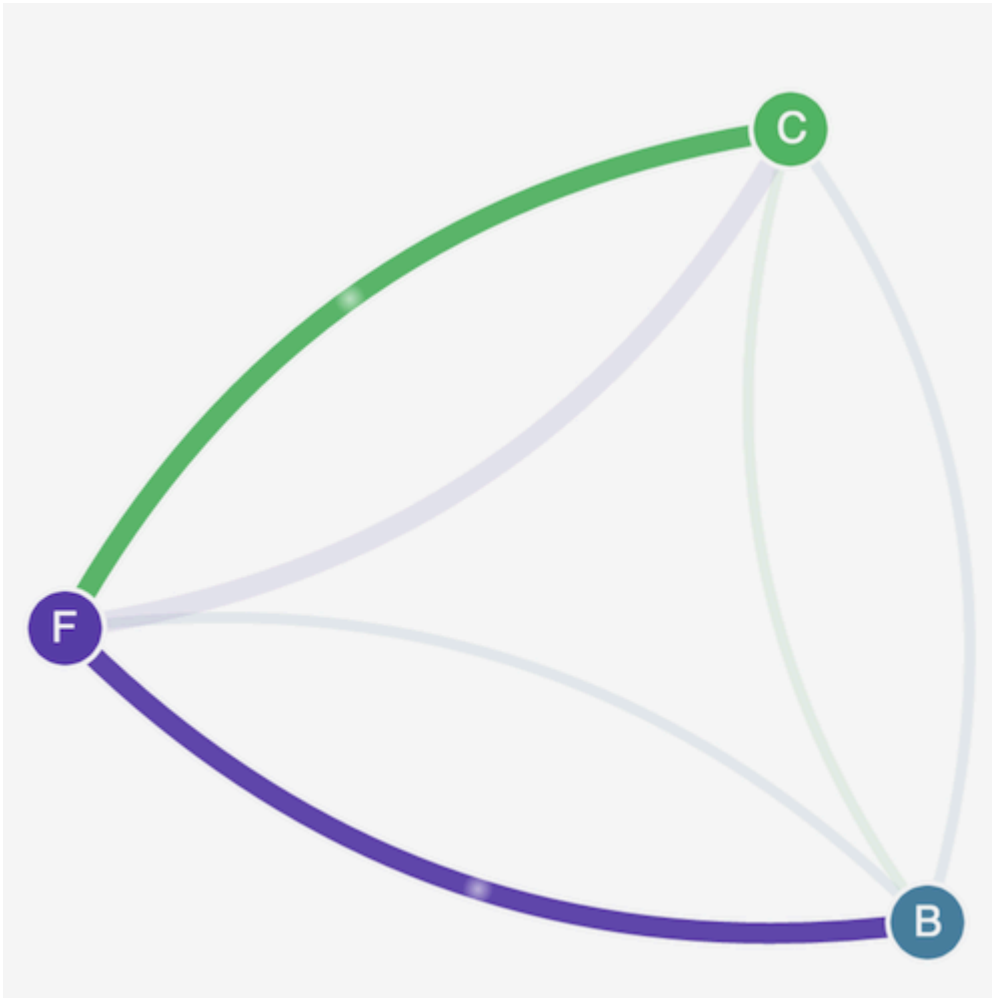
11. Terapkan kebijakan jaringan berikut untuk mengizinkan lalu lintas dari klien ke layanan front-end:

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: stars
  name: frontend-policy
spec:
  podSelector:
    matchLabels:
      role: frontend
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              role: client
      ports:
        - protocol: TCP
```



```
port: 80
```

12. Segarkan peramban Anda. Anda melihat bahwa klien dapat berkomunikasi dengan layanan front-end. Layanan front-end masih dapat berkomunikasi dengan layanan back-end.



13.(Opsional) Setelah selesai dengan demo, Anda dapat menghapus sumber dayanya.

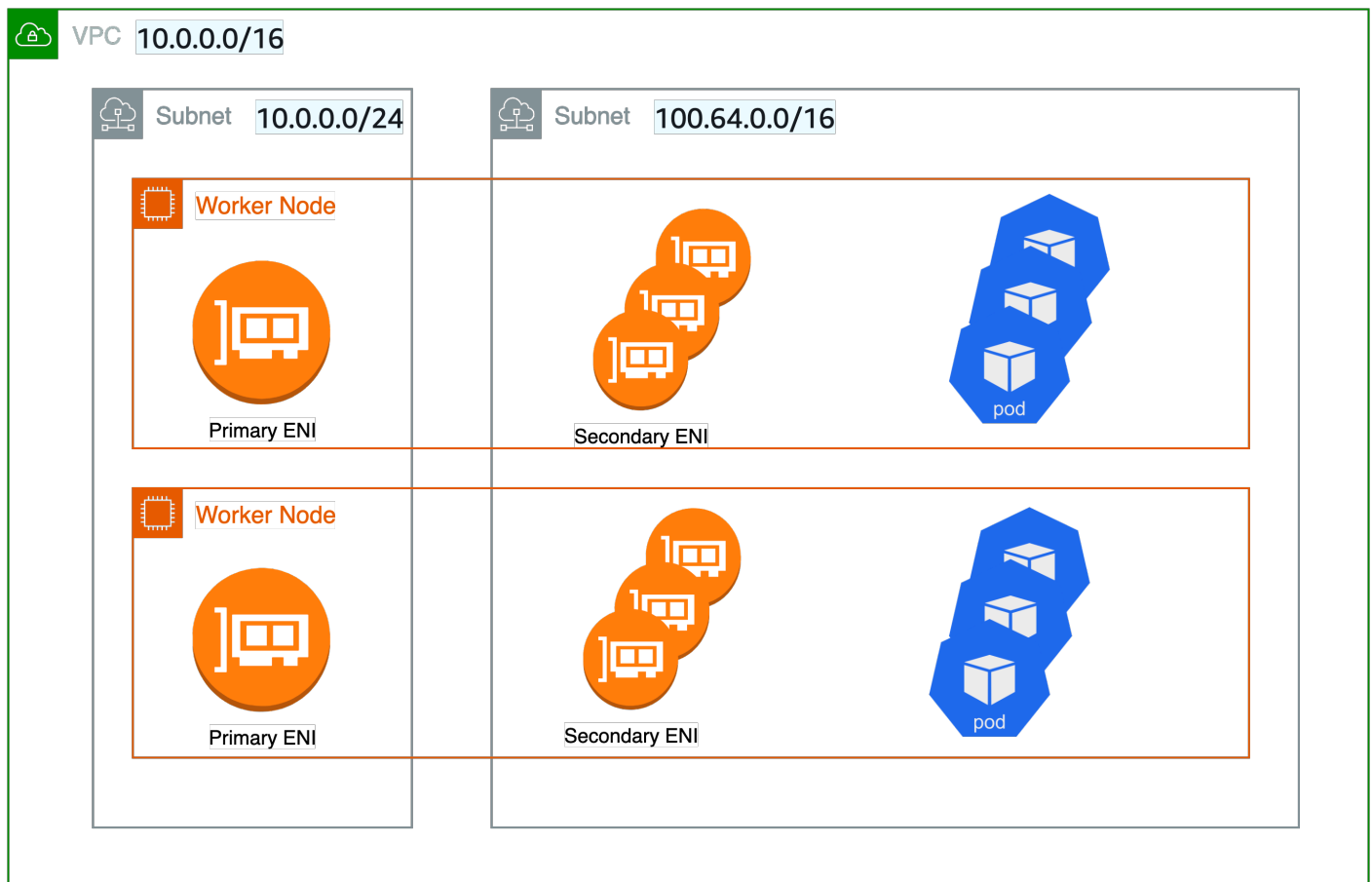
```
kubectl delete -f https://raw.githubusercontent.com/aws-samples/eks-workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-policies/calico/stars_policy_demo/create_resources.files/client.yaml
kubectl delete -f https://raw.githubusercontent.com/aws-samples/eks-workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-policies/calico/stars_policy_demo/create_resources.files/frontend.yaml
kubectl delete -f https://raw.githubusercontent.com/aws-samples/eks-workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-policies/calico/stars_policy_demo/create_resources.files/backend.yaml
```

```
kubectl delete -f https://raw.githubusercontent.com/aws-samples/eks-workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-policies/calico/stars_policy_demo/create_resources.files/management-ui.yaml
kubectl delete -f https://raw.githubusercontent.com/aws-samples/eks-workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-policies/calico/stars_policy_demo/create_resources.files/namespace.yaml
```

Bahkan setelah menghapus sumber daya, masih ada titik akhir kebijakan jaringan pada node yang mungkin mengganggu jaringan di cluster Anda dengan cara yang tidak terduga. Satu-satunya cara pasti untuk menghapus aturan ini adalah dengan me-reboot node atau menghentikan semua node dan mendaur ulangnya. Untuk mengakhiri semua simpul, atur jumlah Grup Auto Scaling yang diinginkan ke 0, lalu buat cadangan ke jumlah yang diinginkan, atau cukup hentikan simpul.

Menerapkan Pod di subnet alternatif dengan jaringan khusus

Berlaku untuk: Node IPv4 Fargate Linux, node Linux dengan instance Amazon EC2



Secara default, ketika plugin Amazon VPC CNI untuk Kubernetes membuat antarmuka [jaringan elastis sekunder \(antarmuka jaringan\)](#) untuk EC2 node Amazon Anda, itu membuatnya dalam subnet yang sama dengan antarmuka jaringan utama node. Ini juga mengaitkan kelompok keamanan yang sama ke antarmuka jaringan sekunder yang terkait dengan antarmuka jaringan utama. Untuk satu atau beberapa alasan berikut, Anda mungkin ingin plugin membuat antarmuka jaringan sekunder di subnet yang berbeda atau ingin mengaitkan grup keamanan yang berbeda ke antarmuka jaringan sekunder, atau keduanya:

- Ada sejumlah IPv4 alamat terbatas yang tersedia di subnet tempat antarmuka jaringan utama berada. Ini mungkin membatasi jumlah Pod yang dapat Anda buat di subnet. Dengan menggunakan subnet yang berbeda untuk antarmuka jaringan sekunder, Anda dapat menambah jumlah IPv4 alamat yang tersedia untuk Pod.
- Untuk alasan keamanan, Pod Anda mungkin perlu menggunakan subnet atau grup keamanan yang berbeda dari antarmuka jaringan utama node.
- Node dikonfigurasi dalam subnet publik, dan Anda ingin menempatkan Pod di subnet pribadi. Tabel rute yang terkait dengan subnet publik mencakup rute ke gateway internet. Tabel rute yang terkait dengan subnet pribadi tidak menyertakan rute ke gateway internet.

Tip

Anda juga dapat menambahkan subnet baru atau yang sudah ada langsung ke Amazon EKS Cluster Anda, tanpa menggunakan jaringan khusus. Untuk informasi selengkapnya, lihat [the section called “Tambahkan Subnet VPC yang ada ke cluster Amazon EKS dari konsol manajemen”](#).

Pertimbangan

Berikut ini adalah pertimbangan untuk menggunakan fitur ini.

- Dengan mengaktifkan jaringan khusus, tidak ada alamat IP yang ditetapkan ke antarmuka jaringan utama yang ditetapkan ke Pod. Hanya alamat IP dari antarmuka jaringan sekunder yang ditetapkan ke Pod.
- Jika klaster Anda menggunakan IPv6 keluarga, Anda tidak dapat menggunakan jaringan khusus.
- Jika Anda berencana untuk menggunakan jaringan khusus hanya untuk membantu mengurangi kelelahan IPv4 alamat, Anda dapat membuat klaster menggunakan keluarga sebagai gantinya. IPv6 Untuk informasi selengkapnya, lihat [the section called “IPv6”](#).

- Meskipun Pod yang di-deploy ke subnet yang ditentukan untuk antarmuka jaringan sekunder dapat menggunakan subnet dan grup keamanan yang berbeda dari antarmuka jaringan utama node, subnet dan grup keamanan harus berada dalam VPC yang sama dengan node.
- Untuk Fargate, subnet dikendalikan melalui profil Fargate. Untuk informasi selengkapnya, lihat [the section called “Tentukan profil”](#).

Sesuaikan antarmuka jaringan sekunder di node Amazon EKS

Selesaikan yang berikut ini sebelum Anda memulai tutorial:

- Tinjau pertimbangannya
- Keakraban dengan bagaimana plugin Amazon VPC CNI untuk Kubernetes membuat antarmuka jaringan sekunder dan menetapkan alamat IP ke Pod. Untuk informasi lebih lanjut, lihat [Alokasi ENI](#) di GitHub.
- Versi 2.12.3 atau yang lebih baru atau versi 1.27.160 atau yang lebih baru dari AWS Command Line Interface (AWS CLI) diinstal dan dikonfigurasi pada perangkat Anda atau AWS CloudShell. Untuk memeriksa versi Anda saat ini, gunakan `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Package manager seperti `yum` atau `apt-get`, atau Homebrew untuk macOS seringkali merupakan beberapa versi di belakang versi terbaru CLI AWS. Untuk menginstal versi terbaru, lihat [Menginstal](#) dan [Konfigurasi cepat dengan aws configure](#) di Panduan Pengguna Antarmuka Baris AWS Perintah. Versi AWS CLI yang diinstal AWS CloudShell mungkin juga beberapa versi di belakang versi terbaru. Untuk memperbaruinya, lihat [Menginstal AWS CLI ke direktori home Anda](#) di AWS CloudShell Panduan Pengguna.
- Alat baris perintah `kubectl` diinstal pada perangkat Anda atau AWS CloudShell. Untuk menginstal atau memutakhirkan `kubectl`, lihat [the section called “Siapkan kubectl dan eksctl”](#).
- Kami menyarankan Anda menyelesaikan langkah-langkah dalam topik ini di shell Bash. Jika Anda tidak menggunakan shell Bash, beberapa perintah skrip seperti karakter kelanjutan baris dan cara variabel diatur dan digunakan memerlukan penyesuaian untuk shell Anda. Selain itu, aturan mengutip dan melarikan diri untuk shell Anda mungkin berbeda. Untuk informasi selengkapnya, lihat [Menggunakan tanda kutip dengan string di AWS CLI](#) di Panduan Pengguna Antarmuka Baris AWS Perintah.

Untuk tutorial ini, kami sarankan menggunakan nilai contoh, kecuali jika dicatat untuk menggantikannya. Anda dapat mengganti nilai contoh apa pun saat menyelesaikan langkah-langkah untuk kluster produksi. Kami merekomendasikan untuk menyelesaikan semua langkah di terminal

yang sama. Ini karena variabel diatur dan digunakan di seluruh langkah dan tidak akan ada di terminal yang berbeda.

Perintah dalam topik ini diformat menggunakan konvensi yang tercantum dalam [Menggunakan contoh CLI AWS](#). Jika Anda menjalankan perintah dari baris perintah terhadap sumber daya yang berada di AWS Wilayah yang berbeda dari AWS Wilayah default yang ditentukan dalam [profil AWS CLI](#) yang Anda gunakan, maka Anda perlu menambahkan `--region us-west-2` perintah, menggantinya `us-west-2` dengan wilayah Anda AWS .

Saat Anda ingin menerapkan jaringan khusus ke kluster produksi Anda, lewati ke [the section called “Langkah 2: Konfigurasi VPC Anda”](#).

Langkah 1: Buat tes VPC dan cluster

Prosedur berikut membantu Anda membuat pengujian VPC dan cluster dan mengkonfigurasi jaringan khusus untuk cluster itu. Kami tidak menyarankan penggunaan kluster pengujian untuk beban kerja produksi karena beberapa fitur yang tidak terkait yang mungkin Anda gunakan pada kluster produksi tidak tercakup dalam topik ini. Untuk informasi selengkapnya, lihat [the section called “Membuat kluster”](#).

1. Jalankan perintah berikut untuk menentukan `account_id` variabel.

```
account_id=$(aws sts get-caller-identity --query Account --output text)
```

2. Buat sebuah VPC.

- a. Jika Anda menerapkan ke sistem pengujian, buat VPC menggunakan template Amazon AWS CloudFormation EKS.

```
aws cloudformation create-stack --stack-name my-eks-custom-networking-vpc \
  --template-url https://s3.us-west-2.amazonaws.com/amazon-eks/
cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml \
  --parameters ParameterKey=VpcBlock,ParameterValue=192.168.0.0/24 \
  ParameterKey=PrivateSubnet01Block,ParameterValue=192.168.0.64/27 \
  ParameterKey=PrivateSubnet02Block,ParameterValue=192.168.0.96/27 \
  ParameterKey=PublicSubnet01Block,ParameterValue=192.168.0.0/27 \
  ParameterKey=PublicSubnet02Block,ParameterValue=192.168.0.32/27
```

- b. AWS CloudFormation Tumpukan membutuhkan waktu beberapa menit untuk membuatnya. Untuk memeriksa status penyebaran tumpukan, jalankan perintah berikut.

```
aws cloudformation describe-stacks --stack-name my-eks-custom-networking-vpc --
query Stacks\[\\].StackStatus --output text
```

Jangan melanjutkan ke langkah berikutnya sampai output dari perintah tersebut `CREATE_COMPLETE`.

- c. Tentukan variabel dengan nilai subnet pribadi yang IDs dibuat oleh template.

```
subnet_id_1=$(aws cloudformation describe-stack-resources --stack-name my-eks-
custom-networking-vpc \
  --query "StackResources[?
LogicalResourceId=='PrivateSubnet01'].PhysicalResourceId" --output text)
subnet_id_2=$(aws cloudformation describe-stack-resources --stack-name my-eks-
custom-networking-vpc \
  --query "StackResources[?
LogicalResourceId=='PrivateSubnet02'].PhysicalResourceId" --output text)
```

- d. Tentukan variabel dengan Availability Zones dari subnet yang diambil pada langkah sebelumnya.

```
az_1=$(aws ec2 describe-subnets --subnet-ids $subnet_id_1 --query
'Subnets[*].AvailabilityZone' --output text)
az_2=$(aws ec2 describe-subnets --subnet-ids $subnet_id_2 --query
'Subnets[*].AvailabilityZone' --output text)
```

3. Buat peran IAM cluster.

- a. Jalankan perintah berikut untuk membuat file JSON kebijakan kepercayaan IAM.

```
cat >eks-cluster-role-trust-policy.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

EOF

- b. Buat peran IAM cluster Amazon EKS. Jika perlu, kata pengantar eks-cluster-role-trust-policy.json dengan jalur di komputer Anda tempat Anda menulis file pada langkah sebelumnya. Perintah tersebut mengaitkan kebijakan kepercayaan yang Anda buat pada langkah sebelumnya dengan peran tersebut. Untuk membuat peran IAM, [prinsipal IAM](#) yang membuat peran harus diberi iam:CreateRole tindakan (izin).

```
aws iam create-role --role-name myCustomNetworkingAmazonEKSClusterRole --assume-role-policy-document file://"eks-cluster-role-trust-policy.json"
```

- c. Lampirkan kebijakan terkelola Amazon EKS bernama [EKSClusterKebijakan Amazon](#) ke peran tersebut. Untuk melampirkan kebijakan IAM ke kepala sekolah [IAM, prinsipal](#) yang melampirkan kebijakan harus diberikan salah satu tindakan IAM berikut (izin): atau. iam:AttachUserPolicy iam:AttachRolePolicy

```
aws iam attach-role-policy --policy-arn arn:aws: iam::aws:policy/AmazonEKSClusterPolicy --role-name myCustomNetworkingAmazonEKSClusterRole
```

4. Buat kluster Amazon EKS dan konfigurasi perangkat Anda untuk berkomunikasi dengannya.
 - a. Buat sebuah klaster.

```
aws eks create-cluster --name my-custom-networking-cluster \
  --role-arn arn:aws: iam::$account_id:role/
myCustomNetworkingAmazonEKSClusterRole \
  --resources-vpc-config subnetIds=$subnet_id_1,"$subnet_id_2
```

Note

Anda mungkin menerima kesalahan bahwa salah satu Availability Zone dalam permintaan Anda tidak memiliki kapasitas yang cukup untuk membuat klaster Amazon EKS. Jika hal ini terjadi, output galat berisi Availability Zones yang dapat mendukung klaster baru. Cobalah untuk kembali membuat klaster dengan setidaknya dua subnet yang terletak di Availability Zones yang didukung untuk akun Anda. Untuk informasi selengkapnya, lihat [the section called “Kapasitas tidak mencukupi”](#).

- b. Cluster membutuhkan waktu beberapa menit untuk membuatnya. Untuk memeriksa status penyebaran cluster, jalankan perintah berikut.

```
aws eks describe-cluster --name my-custom-networking-cluster --query
cluster.status
```

Jangan melanjutkan ke langkah berikutnya sampai output dari perintah tersebut "ACTIVE".

- c. Konfigurasi `kubectl` untuk berkomunikasi dengan cluster Anda.

```
aws eks update-kubeconfig --name my-custom-networking-cluster
```

Langkah 2: Konfigurasi VPC Anda

Tutorial ini membutuhkan VPC yang dibuat di [the section called “Langkah 1: Buat tes VPC dan cluster”](#). Untuk cluster produksi, sesuaikan langkah-langkah yang sesuai untuk VPC Anda dengan mengganti semua nilai contoh dengan milik Anda sendiri.

1. Konfirmasikan bahwa plugin Amazon VPC CNI Anda yang saat ini diinstal untuk Kubernetes adalah versi terbaru. Untuk menentukan versi terbaru untuk jenis add-on Amazon EKS dan memperbarui versi Anda ke sana, lihat [the section called “Perbarui add-on”](#). Untuk menentukan versi terbaru untuk jenis add-on yang dikelola sendiri dan memperbarui versi Anda ke sana, lihat [the section called “Amazon VPC CNI”](#).
2. Ambil ID VPC cluster Anda dan simpan dalam variabel untuk digunakan di langkah selanjutnya.

```
vpc_id=$(aws eks describe-cluster --name my-custom-networking-cluster --query
"cluster.resourcesVpcConfig.vpcId" --output text)
```

3. Kaitkan blok Classless Inter-Domain Routing (CIDR) tambahan dengan VPC klaster Anda. Blok CIDR tidak dapat tumpang tindih dengan blok CIDR terkait yang ada.
 - a. Lihat blok CIDR saat ini yang terkait dengan VPC Anda.

```
aws ec2 describe-vpcs --vpc-ids $vpc_id \
  --query 'Vpcs[*].CidrBlockAssociationSet[*].{CIDRBlock: CidrBlock, State:
  CidrBlockState.State}' --out table
```

Contoh output adalah sebagai berikut.

```
-----
|           DescribeVpcs           |
+-----+-----+
|
```


CIDRBlock	State
192.168.0.0/24	associated

- b. Kaitkan blok CIDR tambahan ke VPC Anda. Ganti nilai blok CIDR dalam perintah berikut. Untuk informasi selengkapnya, lihat [Mengaitkan blok IPv4 CIDR tambahan dengan VPC Anda](#) di Panduan Pengguna Amazon VPC.

```
aws ec2 associate-vpc-cidr-block --vpc-id $vpc_id --cidr-block 192.168.1.0/24
```

- c. Konfirmasikan bahwa blok baru dikaitkan.

```
aws ec2 describe-vpcs --vpc-ids $vpc_id --query
'Vpcs[*].CidrBlockAssociationSet[*].{CIDRBlock: CidrBlock, State:
CidrBlockState.State}' --out table
```

Contoh output adalah sebagai berikut.

DescribeVpcs	
CIDRBlock	State
192.168.0.0/24	associated
192.168.1.0/24	associated

Jangan lanjutkan ke langkah berikutnya sampai blok CIDR baru Anda selesaiState. associated

4. Buat subnet sebanyak yang ingin Anda gunakan di setiap Availability Zone tempat subnet Anda ada. Tentukan blok CIDR yang ada di dalam blok CIDR yang Anda kaitkan dengan VPC Anda di langkah sebelumnya.
 - a. Buat subnet baru. Ganti nilai blok CIDR dalam perintah berikut. Subnet harus dibuat di blok CIDR VPC yang berbeda dari subnet yang ada, tetapi di Availability Zone yang sama dengan subnet yang ada. Dalam contoh ini, satu subnet dibuat di blok CIDR baru di setiap Availability Zone tempat subnet pribadi saat ini ada. Subnet IDs yang dibuat disimpan dalam variabel untuk digunakan dalam langkah-langkah selanjutnya. NameNilai cocok dengan nilai yang ditetapkan ke subnet yang dibuat menggunakan template Amazon EKS VPC pada langkah sebelumnya. Nama tidak diperlukan. Anda dapat menggunakan nama yang berbeda.

```
new_subnet_id_1=$(aws ec2 create-subnet --vpc-id $vpc_id --availability-zone $az_1
--cidr-block 192.168.1.0/27 \
    --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=my-eks-custom-
networking-vpc-PrivateSubnet01},{Key=kubernetes.io/role/internal-elb,Value=1}]' \
    --query Subnet.SubnetId --output text)
new_subnet_id_2=$(aws ec2 create-subnet --vpc-id $vpc_id --availability-zone $az_2
--cidr-block 192.168.1.32/27 \
    --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=my-eks-custom-
networking-vpc-PrivateSubnet02},{Key=kubernetes.io/role/internal-elb,Value=1}]' \
    --query Subnet.SubnetId --output text)
```

Important

Secara default, subnet baru Anda secara implisit terkait dengan tabel rute utama VPC Anda. Tabel rute ini memungkinkan komunikasi antara semua sumber daya yang digunakan di VPC. Namun, itu tidak memungkinkan komunikasi dengan sumber daya yang memiliki alamat IP yang berada di luar blok CIDR yang terkait dengan VPC Anda. Anda dapat mengaitkan tabel rute Anda sendiri ke subnet Anda untuk mengubah perilaku ini. Untuk informasi selengkapnya, lihat [Tabel rute subnet](#) di Panduan Pengguna Amazon VPC.

b. Lihat subnet saat ini di VPC Anda.

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=$vpc_id" \
    --query 'Subnets[*].{SubnetId: SubnetId,AvailabilityZone:
AvailabilityZone,CidrBlock: CidrBlock}' \
    --output table
```

Contoh output adalah sebagai berikut.

```
-----
|                               DescribeSubnets                               |
+-----+-----+-----+-----+
| AvailabilityZone | CidrBlock | SubnetId |
+-----+-----+-----+-----+
| us-west-2d      | 192.168.0.0/27 | subnet-example1 |
| us-west-2a      | 192.168.0.32/27 | subnet-example2 |
| us-west-2a      | 192.168.0.64/27 | subnet-example3 |
| us-west-2d      | 192.168.0.96/27 | subnet-example4 |
+-----+-----+-----+-----+
```

us-west-2a	192.168.1.0/27	subnet-example5
us-west-2d	192.168.1.32/27	subnet-example6

Anda dapat melihat subnet di blok 192.168.1.0 CIDR yang Anda buat berada di Availability Zone yang sama dengan subnet di blok CIDR. 192.168.0.0

Langkah 3: Konfigurasi sumber daya Kubernetes

1. Atur variabel `AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG` lingkungan ke `true` dalam `aws-node DaemonSet`.

```
kubectl set env daemonset aws-node -n kube-system
AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG=true
```

2. Ambil ID [grup keamanan klaster](#) Anda dan simpan dalam variabel untuk digunakan pada langkah berikutnya. Amazon EKS secara otomatis membuat grup keamanan ini saat Anda membuat klaster.

```
cluster_security_group_id=$(aws eks describe-cluster --name my-custom-networking-cluster --query cluster.resourcesVpcConfig.clusterSecurityGroupId --output text)
```

3. Buat sumber daya `ENIConfig` khusus untuk setiap subnet yang ingin Anda gunakan untuk menerapkan Pod.
 - a. Buat file unik untuk setiap konfigurasi antarmuka jaringan.

Perintah berikut membuat `ENIConfig` file terpisah untuk dua subnet yang dibuat pada langkah sebelumnya. Nilai untuk `name` harus unik. Namanya sama dengan Availability Zone tempat subnet berada. Grup keamanan cluster ditugaskan ke file `ENIConfig`.

```
cat >$az_1.yaml <<EOF
apiVersion: crd.k8s.amazonaws.com/v1alpha1
kind: ENIConfig
metadata:
  name: $az_1
spec:
  securityGroups:
    - $cluster_security_group_id
  subnet: $new_subnet_id_1
```

EOF

```
cat >$az_2.yaml <<EOF
apiVersion: crd.k8s.amazonaws.com/v1alpha1
kind: ENIConfig
metadata:
  name: $az_2
spec:
  securityGroups:
    - $cluster_security_group_id
  subnet: $new_subnet_id_2
EOF
```

Untuk cluster produksi, Anda dapat membuat perubahan berikut pada perintah sebelumnya:

- Ganti `$cluster_security_group_id` dengan ID grup [keamanan](#) yang ada yang ingin Anda gunakan untuk masing-masing. ENIConfig
- Sebaiknya beri nama Anda ENIConfigs sama dengan Availability Zone yang akan Anda gunakan, bila memungkinkan. ENIConfig Anda mungkin perlu menggunakan nama yang berbeda untuk nama Availability Zone karena berbagai alasan. ENIConfigs Misalnya, jika Anda memiliki lebih dari dua subnet di Availability Zone yang sama dan ingin menggunakan keduanya dengan jaringan kustom, maka Anda memerlukan beberapa ENIConfigs untuk Availability Zone yang sama. Karena masing-masing ENIConfig memerlukan nama unik, Anda tidak dapat memberi nama lebih dari satu nama Anda ENIConfigs menggunakan nama Availability Zone.

Jika ENIConfig nama Anda tidak semua sama dengan nama Availability Zone, maka ganti `$az_1` dan `$az_2` dengan nama Anda sendiri di perintah sebelumnya dan [beri anotasi node Anda](#) dengan yang nanti dalam tutorial ini. ENIConfig

 Note

Jika Anda tidak menentukan grup keamanan yang valid untuk digunakan dengan cluster produksi dan Anda menggunakan:

- versi 1.8.0 atau yang lebih baru dari plugin Amazon VPC CNI untuk Kubernetes, kemudian kelompok keamanan yang terkait dengan antarmuka elastic network utama node digunakan.
- versi plugin Amazon VPC CNI untuk Kubernetes yang lebih awal dari 1.8.0, maka grup keamanan default untuk VPC ditetapkan ke antarmuka jaringan sekunder.

⚠ Important

- `AWS_VPC_K8S_CNI_EXTERNALSNAT=false` adalah pengaturan default dalam konfigurasi untuk plugin Amazon VPC CNI untuk Kubernetes. Jika Anda menggunakan pengaturan default, lalu lintas yang ditujukan untuk alamat IP yang tidak berada dalam salah satu blok CIDR yang terkait dengan VPC Anda menggunakan grup keamanan dan subnet antarmuka jaringan utama node Anda. Subnet dan grup keamanan yang ditentukan dalam Anda ENIConfigs yang digunakan untuk membuat antarmuka jaringan sekunder tidak digunakan untuk lalu lintas ini. Untuk informasi selengkapnya tentang pengaturan ini, lihat [the section called “Lalu lintas keluar”](#).
- Jika Anda juga menggunakan grup keamanan untuk Pod, grup keamanan yang ditentukan dalam sebuah SecurityGroupPolicy akan digunakan sebagai ganti grup keamanan yang ditentukan dalam ENIConfigs. Untuk informasi selengkapnya, lihat [the section called “Grup keamanan untuk Pod”](#).

- b. Terapkan setiap file sumber daya kustom yang Anda buat ke cluster Anda dengan perintah berikut.

```
kubectl apply -f $az_1.yaml
kubectl apply -f $az_2.yaml
```

4. Konfirmasikan bahwa Anda ENIConfigs telah dibuat.

```
kubectl get ENIConfigs
```

Contoh output adalah sebagai berikut.

NAME	AGE
us-west-2a	117s
us-west-2d	105s

5. Jika Anda mengaktifkan jaringan khusus pada klaster produksi dan menamai ENIConfigs sesuatu selain Availability Zone tempat Anda menggunakannya, lewati ke [langkah berikutnya](#) untuk menerapkan node Amazon EC2 .

Aktifkan Kubernetes untuk secara otomatis menerapkan Availability Zone ke EC2 node Amazon baru yang dibuat di klaster Anda. ENIConfig

- a. Untuk test cluster dalam tutorial ini, lompat ke [langkah berikutnya](#).

Untuk cluster produksi, periksa untuk melihat apakah anotasi dengan kunci `k8s.amazonaws.com/eniConfig` untuk variabel [ENI_CONFIG_ANNOTATION_DEF](#) lingkungan ada dalam spesifikasi container untuk `aws-node` DaemonSet

```
kubectl describe daemonset aws-node -n kube-system | grep
ENI_CONFIG_ANNOTATION_DEF
```

Jika output dikembalikan, anotasi ada. Jika tidak ada output yang dikembalikan, maka variabel tidak diatur. Untuk cluster produksi, Anda dapat menggunakan pengaturan ini atau pengaturan di langkah berikut. Jika Anda menggunakan pengaturan ini, itu akan mengganti pengaturan pada langkah berikut. Dalam tutorial ini, pengaturan pada langkah berikutnya digunakan.

- b. Perbarui `aws-node` DaemonSet untuk menerapkan Availability Zone secara otomatis ke EC2 node Amazon baru yang dibuat di cluster Anda. `ENIConfig`

```
kubectl set env daemonset aws-node -n kube-system
ENI_CONFIG_LABEL_DEF=topology.kubernetes.io/zone
```

Langkah 4: Menyebarkan node Amazon EC2

1. Buat peran IAM node.

- a. Jalankan perintah berikut untuk membuat file JSON kebijakan kepercayaan IAM.

```
cat >node-role-trust-relationship.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

- b. Buat peran IAM dan simpan Amazon Resource Name (ARN) yang dikembalikan dalam variabel untuk digunakan di langkah selanjutnya.

```
node_role_arn=$(aws iam create-role --role-name myCustomNetworkingNodeRole --
assume-role-policy-document file://"node-role-trust-relationship.json" \
--query Role.Arn --output text)
```

- c. Lampirkan tiga kebijakan yang dikelola IAM yang diperlukan ke peran IAM.

```
aws iam attach-role-policy \
--policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy \
--role-name myCustomNetworkingNodeRole
aws iam attach-role-policy \
--policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
--role-name myCustomNetworkingNodeRole
aws iam attach-role-policy \
--policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
--role-name myCustomNetworkingNodeRole
```

Important

Untuk kesederhanaan dalam tutorial ini, kebijakan [AmazonEKS_CNI_Policy dilampirkan ke peran](#) IAM node. Namun, dalam cluster produksi, kami merekomendasikan untuk melampirkan kebijakan ke peran IAM terpisah yang hanya digunakan dengan plugin Amazon VPC CNI untuk Kubernetes. Untuk informasi selengkapnya, lihat [the section called “Konfigurasi untuk IRSA”](#).

2. Buat salah satu jenis grup node berikut. Untuk menentukan jenis instance yang ingin Anda terapkan, lihat [the section called “Jenis EC2 instans Amazon”](#). Untuk tutorial ini, lengkapi template Managed, Without a launch atau dengan template peluncuran tanpa opsi yang ditentukan ID AMI. Jika Anda akan menggunakan grup node untuk beban kerja produksi, maka sebaiknya Anda membiasakan diri dengan semua grup node [terkelola dan opsi grup node](#) yang [dikelola sendiri sebelum menerapkan grup node](#).
- Dikelola - Terapkan grup node Anda menggunakan salah satu opsi berikut:
 - Tanpa template peluncuran atau dengan template peluncuran tanpa ID AMI yang ditentukan - Jalankan perintah berikut. Untuk tutorial ini, gunakan nilai contoh. Untuk grup node produksi, ganti semua nilai contoh dengan milik Anda sendiri. Nama grup node tidak boleh lebih dari

63 karakter. Itu harus dimulai dengan huruf atau digit, tetapi juga dapat menyertakan tanda hubung dan garis bawah untuk karakter yang tersisa.

```
aws eks create-nodegroup --cluster-name my-custom-networking-cluster --nodegroup-name my-nodegroup \
    --subnets $subnet_id_1 $subnet_id_2 --instance-types t3.medium --node-role $node_role_arn
```

- Dengan template peluncuran dengan ID AMI yang ditentukan
 - A. Tentukan jumlah maksimum Pod yang direkomendasikan Amazon EKS untuk node Anda. Ikuti petunjuk di [Amazon EKS Pod maksimum yang direkomendasikan untuk setiap jenis EC2 instans Amazon](#), tambahkan `--cni-custom-networking-enabled` ke langkah 3 dalam topik tersebut. Perhatikan output untuk digunakan pada langkah berikutnya.
 - B. Di template peluncuran Anda, tentukan ID AMI Amazon EKS yang dioptimalkan, atau AMI kustom yang dibuat dari AMI Amazon EKS yang dioptimalkan, lalu [terapkan grup node menggunakan templat peluncuran](#) dan berikan data pengguna berikut di template peluncuran. Data pengguna ini meneruskan argumen ke dalam `bootstrap.sh` file. Untuk informasi lebih lanjut tentang file bootstrap, lihat [bootstrap.sh](#) di GitHub. Anda dapat mengganti `20` dengan nilai dari langkah sebelumnya (disarankan) atau nilai Anda sendiri.

```
/etc/eks/bootstrap.sh my-custom-networking-cluster --use-max-pods false --
kubelet-extra-args '--max-pods=20'
```

Jika Anda telah membuat AMI khusus yang tidak dibangun dari AMI Amazon EKS yang dioptimalkan, maka Anda perlu membuat konfigurasi sendiri.

- Dikelola sendiri
 - i. Tentukan jumlah maksimum Pod yang direkomendasikan Amazon EKS untuk node Anda. Ikuti instruksi di [the section called “Amazon EKS merekomendasikan Pod maksimum untuk setiap jenis EC2 instans Amazon”](#), tambahkan `--cni-custom-networking-enabled` ke langkah 3 dalam topik itu. Perhatikan output untuk digunakan pada langkah berikutnya.
 - ii. Menyebarkan grup simpul menggunakan instruksi di [the section called “Amazon Linux”](#). Tentukan teks berikut untuk `BootstrapArguments` parameter. Anda dapat mengganti `20` dengan nilai dari langkah sebelumnya (disarankan) atau nilai Anda sendiri.

```
--use-max-pods false --kubelet-extra-args '--max-pods=20'
```


Note

Jika Anda ingin node dalam kluster produksi mendukung jumlah Pod yang jauh lebih tinggi, jalankan skrip [the section called “Amazon EKS merekomendasikan Pod maksimum untuk setiap jenis EC2 instans Amazon”](#) lagi. Juga, tambahkan `--cni-prefix-delegation-enabled` opsi ke perintah. Misalnya, 110 dikembalikan untuk tipe `m5.large` instance. Untuk petunjuk tentang cara mengaktifkan kemampuan ini, lihat [the section called “Meningkatkan alamat IP”](#). Anda dapat menggunakan kemampuan ini dengan jaringan khusus.

3. Pembuatan grup node membutuhkan waktu beberapa menit. Anda dapat memeriksa status pembuatan grup node terkelola dengan perintah berikut.

```
aws eks describe-nodegroup --cluster-name my-custom-networking-cluster --nodegroup-name my-nodegroup --query nodegroup.status --output text
```

Jangan melanjutkan ke langkah berikutnya sampai output yang dikembalikan `ACTIVE`.

4. Untuk tutorialnya, Anda dapat melewati langkah ini.

Untuk cluster produksi, jika Anda tidak memberi nama yang sama `ENIConfigs` dengan Availability Zone yang Anda gunakan untuk mereka, maka Anda harus membubuhi keterangan node Anda dengan `ENIConfig` nama yang harus digunakan dengan node. Langkah ini tidak diperlukan jika Anda hanya memiliki satu subnet di setiap Availability Zone dan Anda menamai Anda `ENIConfigs` dengan nama yang sama dengan Availability Zones Anda. [Ini karena plugin Amazon VPC CNI untuk Kubernetes secara otomatis mengaitkan yang benar `ENIConfig` dengan node untuk Anda ketika Anda mengaktifkannya untuk melakukannya pada langkah sebelumnya.](#)

- a. Dapatkan daftar node di cluster Anda.

```
kubectl get nodes
```

Contoh output adalah sebagai berikut.

NAME	STATUS	ROLES	AGE	VERSION
ip-192-168-0-126.us-west-2.compute.internal eks-810597c	Ready	<none>	8m49s	v1.22.9-

```
ip-192-168-0-92.us-west-2.compute.internal    Ready    <none>    8m34s    v1.22.9-eks-810597c
```

- b. Tentukan Availability Zone mana setiap node berada. Jalankan perintah berikut untuk setiap node yang dikembalikan pada langkah sebelumnya, mengganti alamat IP berdasarkan output sebelumnya.

```
aws ec2 describe-instances --filters Name=network-interface.private-dns-name,Values=ip-192-168-0-126.us-west-2.compute.internal \
--query 'Reservations[].Instances[].{AvailabilityZone: Placement.AvailabilityZone, SubnetId: SubnetId}'
```

Contoh output adalah sebagai berikut.

```
[
  {
    "AvailabilityZone": "us-west-2d",
    "SubnetId": "subnet-Example5"
  }
]
```

- c. Anotasi setiap node dengan ENIConfig yang Anda buat untuk subnet ID dan Availability Zone. Anda hanya dapat membuat anotasi node dengan satu ENIConfig, meskipun beberapa node dapat dianotasi dengan yang sama. ENIConfig Ganti Contoh nilai dengan nilai Anda sendiri.

```
kubectl annotate node ip-192-168-0-126.us-west-2.compute.internal
k8s.amazonaws.com/eniConfig=EniConfigName1
kubectl annotate node ip-192-168-0-92.us-west-2.compute.internal
k8s.amazonaws.com/eniConfig=EniConfigName2
```

5. Jika Anda memiliki node di kluster produksi dengan menjalankan Pod sebelum beralih menggunakan fitur jaringan kustom, selesaikan tugas-tugas berikut:
- Pastikan Anda memiliki node yang tersedia yang menggunakan fitur jaringan khusus.
 - Cordon dan tiriskan node untuk mematikan Pod dengan anggun. Untuk informasi selengkapnya, lihat [Menguras Node dengan Aman](#) di dokumentasi Kubernetes.
 - Mengakhiri node. Jika node berada dalam grup node terkelola yang ada, Anda dapat menghapus grup node. Jalankan perintah berikut.

```
aws eks delete-nodegroup --cluster-name my-custom-networking-cluster --nodegroup-name my-nodegroup
```

Hanya simpul-simpul baru yang terdaftar dengan label `k8s.amazonaws.com/eniConfig` yang menggunakan fitur jaringan kustom.

6. Konfirmasikan bahwa Pod diberi alamat IP dari blok CIDR yang terkait dengan salah satu subnet yang Anda buat pada langkah sebelumnya.

```
kubectl get pods -A -o wide
```

Contoh output adalah sebagai berikut.

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE	IP
	NODE			NOMINATED	NODE	READINESS
GATES						
kube-system	aws-node-2rk4	1/1	Running	0	7m19s	
	192.168.0.92					ip-192-168-0-92.us-west-2.compute.internal
						<none>
kube-system	aws-node-k96wp	1/1	Running	0	7m15s	
	192.168.0.126					ip-192-168-0-126.us-west-2.compute.internal
						<none>
kube-system	coredns-657694c6f4-smcgr	1/1	Running	0	56m	
	192.168.1.23					ip-192-168-0-92.us-west-2.compute.internal
						<none>
kube-system	coredns-657694c6f4-stwv9	1/1	Running	0	56m	
	192.168.1.28					ip-192-168-0-92.us-west-2.compute.internal
						<none>
kube-system	kube-proxy-jgshq	1/1	Running	0	7m19s	
	192.168.0.92					ip-192-168-0-92.us-west-2.compute.internal
						<none>
kube-system	kube-proxy-wx9vk	1/1	Running	0	7m15s	
	192.168.0.126					ip-192-168-0-126.us-west-2.compute.internal
						<none>

Anda dapat melihat bahwa Pod coredns diberi alamat IP dari blok 192.168.1.0 CIDR yang Anda tambahkan ke VPC Anda. Tanpa jaringan khusus, mereka akan diberi alamat dari blok 192.168.0.0 CIDR, karena itu adalah satu-satunya blok CIDR yang awalnya terkait dengan VPC.

Jika sebuah Pod spec berisi `hostNetwork=true`, itu akan diberikan alamat IP utama dari node tersebut. Itu tidak diberikan alamat dari subnet yang Anda tambahkan. Secara default, nilai ini diatur ke `false`. Nilai ini ditetapkan `true` untuk plugin Amazon VPC CNI kube-proxy dan Amazon untuk Pod Kubernetes (`aws-node`) yang berjalan di klaster Anda. Inilah sebabnya mengapa `aws-node` Pod kube-proxy dan plugin tidak diberi alamat 192.168.1.x pada output sebelumnya. Untuk informasi selengkapnya tentang `hostNetwork` pengaturan Pod, lihat [inti PodSpec v1 di referensi](#) API Kubernetes.

Langkah 5: Hapus sumber daya tutorial

Setelah Anda menyelesaikan tutorial, kami sarankan Anda menghapus sumber daya yang Anda buat. Anda kemudian dapat menyesuaikan langkah-langkah untuk mengaktifkan jaringan khusus untuk klaster produksi.

1. Jika grup node yang Anda buat hanya untuk pengujian, maka hapuslah.

```
aws eks delete-nodegroup --cluster-name my-custom-networking-cluster --nodegroup-name my-nodegroup
```

2. Bahkan setelah output AWS CLI mengatakan bahwa cluster dihapus, proses penghapusan mungkin tidak benar-benar lengkap. Proses penghapusan memakan waktu beberapa menit. Konfirmasikan bahwa itu selesai dengan menjalankan perintah berikut.

```
aws eks describe-nodegroup --cluster-name my-custom-networking-cluster --nodegroup-name my-nodegroup --query nodegroup.status --output text
```

Jangan lanjutkan sampai output yang dikembalikan mirip dengan output berikut.

```
An error occurred (ResourceNotFoundException) when calling the DescribeNodegroup operation: No node group found for name: my-nodegroup.
```

3. Jika grup node yang Anda buat hanya untuk pengujian, maka hapus peran IAM node.

- a. Lepaskan kebijakan dari peran.

```
aws iam detach-role-policy --role-name myCustomNetworkingNodeRole --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy
aws iam detach-role-policy --role-name myCustomNetworkingNodeRole --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
```

```
aws iam detach-role-policy --role-name myCustomNetworkingNodeRole --policy-arn
arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
```

b. Hapus peran.

```
aws iam delete-role --role-name myCustomNetworkingNodeRole
```

4. Hapus klaster .

```
aws eks delete-cluster --name my-custom-networking-cluster
```

Konfirmasikan cluster dihapus dengan perintah berikut.

```
aws eks describe-cluster --name my-custom-networking-cluster --query cluster.status
--output text
```

Ketika output yang mirip dengan berikut ini dikembalikan, cluster berhasil dihapus.

```
An error occurred (ResourceNotFoundException) when calling the DescribeCluster
operation: No cluster found for name: my-custom-networking-cluster.
```

5. Hapus peran IAM cluster.

a. Lepaskan kebijakan dari peran.

```
aws iam detach-role-policy --role-name myCustomNetworkingAmazonEKSClusterRole --
policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy
```

b. Hapus peran.

```
aws iam delete-role --role-name myCustomNetworkingAmazonEKSClusterRole
```

6. Hapus subnet yang Anda buat pada langkah sebelumnya.

```
aws ec2 delete-subnet --subnet-id $new_subnet_id_1
aws ec2 delete-subnet --subnet-id $new_subnet_id_2
```

7. Hapus VPC yang Anda buat.

```
aws cloudformation delete-stack --stack-name my-eks-custom-networking-vpc
```

Tetapkan lebih banyak alamat IP ke node Amazon EKS dengan awalan

Berlaku untuk: Node Linux dan Windows dengan EC2 instans Amazon

Berlaku untuk: Subnet publik dan pribadi

Setiap EC2 instans Amazon mendukung jumlah maksimum antarmuka jaringan elastis dan jumlah maksimum alamat IP yang dapat ditetapkan untuk setiap antarmuka jaringan. Setiap node membutuhkan satu alamat IP untuk setiap antarmuka jaringan. Semua alamat IP lain yang tersedia dapat ditetapkan Pods. Masing-masing Pod membutuhkan alamat IP sendiri. Akibatnya, Anda mungkin memiliki node yang memiliki sumber daya komputasi dan memori yang tersedia, tetapi tidak dapat mengakomodasi tambahan Pods karena node telah kehabisan alamat IP untuk ditetapkan. Pods

Anda dapat meningkatkan jumlah alamat IP yang dapat ditetapkan Pods oleh node dengan menetapkan awalan IP, daripada menetapkan alamat IP sekunder individual ke node Anda. Setiap awalan mencakup beberapa alamat IP. Jika Anda tidak mengonfigurasi kluster untuk penetapan awalan IP, kluster Anda harus membuat lebih banyak panggilan antarmuka pemrograman EC2 aplikasi Amazon (API) untuk mengonfigurasi antarmuka jaringan dan alamat IP yang diperlukan untuk konektivitas Pod. Seiring pertumbuhan cluster ke ukuran yang lebih besar, frekuensi panggilan API ini dapat menyebabkan waktu peluncuran Pod dan instance yang lebih lama. Hal ini mengakibatkan penundaan penskalaan untuk memenuhi permintaan beban kerja yang besar dan runcing, dan menambah biaya dan overhead manajemen karena Anda perlu menyediakan cluster tambahan dan untuk memenuhi persyaratan penskalaan. VPCs Untuk informasi selengkapnya, lihat [Ambang batas skalabilitas Kubernetes pada](#). GitHub

Kompatibilitas dengan plugin Amazon VPC CNI untuk fitur Kubernetes

Anda dapat menggunakan awalan IP dengan fitur-fitur berikut:

- IPv4 Terjemahan Alamat Jaringan Sumber - Untuk informasi lebih lanjut, lihat [the section called “Lalu lintas keluar”](#).
- IPv6 alamat ke cluster, Pod, dan layanan - Untuk informasi selengkapnya, lihat [the section called “IPv6”](#).
- Membatasi lalu lintas menggunakan kebijakan jaringan Kubernetes - Untuk informasi selengkapnya, lihat [the section called “Kebijakan Kubernetes”](#)

Daftar berikut memberikan informasi tentang pengaturan plugin Amazon VPC CNI yang berlaku. Untuk informasi selengkapnya tentang setiap pengaturan, lihat [amazon-vpc-cni-k8s](#) di GitHub.

- WARM_IP_TARGET
- MINIMUM_IP_TARGET
- WARM_PREFIX_TARGET

Pertimbangan

Pertimbangkan hal berikut saat Anda menggunakan fitur ini:

- Setiap jenis EC2 instans Amazon mendukung jumlah maksimum Pod. Jika grup node terkelola Anda terdiri dari beberapa tipe instans, jumlah Pod maksimum terkecil untuk sebuah instance di klaster diterapkan ke semua node dalam klaster.
- Secara default, jumlah maksimum Pods yang dapat Anda jalankan pada node adalah 110, tetapi Anda dapat mengubah nomor itu. Jika Anda mengubah nomor dan memiliki grup node terkelola yang sudah ada, AMI berikutnya atau update template peluncuran grup node Anda akan menghasilkan node baru yang muncul dengan nilai yang diubah.
- Saat beralih dari menetapkan alamat IP ke menetapkan awalan IP, kami menyarankan Anda membuat grup node baru untuk meningkatkan jumlah alamat IP yang tersedia, daripada melakukan penggantian bergulir node yang ada. Menjalankan Pod pada node yang memiliki alamat IP dan prefiks yang ditetapkan dapat menyebabkan ketidakkonsistenan dalam kapasitas alamat IP yang diiklankan, yang berdampak pada beban kerja future pada node. Untuk cara yang disarankan dalam melakukan transisi, lihat [Mengganti semua node selama migrasi dari mode IP Sekunder ke mode Delegasi Awalan atau sebaliknya dalam panduan](#) praktik terbaik Amazon EKS.
- Lingkup grup keamanan berada pada tingkat simpul - Untuk informasi selengkapnya, lihat [Grup keamanan](#).
- Awalan IP yang ditetapkan ke antarmuka jaringan mendukung kepadatan Pod yang tinggi per node dan memiliki waktu peluncuran terbaik.
- Awalan IP dan alamat IP dikaitkan dengan antarmuka jaringan EC2 elastis Amazon standar. Pod yang membutuhkan grup keamanan tertentu diberi alamat IP utama dari antarmuka jaringan cabang. Anda dapat mencampur Pod yang mendapatkan alamat IP, atau alamat IP dari awalan IP dengan Pod yang mendapatkan antarmuka jaringan cabang pada node yang sama.
- Untuk cluster dengan node Linux saja.
 - Setelah Anda mengonfigurasi add-on untuk menetapkan awalan ke antarmuka jaringan, Anda tidak dapat menurunkan versi plugin Amazon VPC CNI untuk add-on Kubernetes ke versi yang 1.9.0 lebih rendah dari (1.10.1atau) tanpa menghapus semua node di semua grup node di cluster Anda.

- Jika Anda juga menggunakan grup keamanan untuk Pod, dengan `POD_SECURITY_GROUP_ENFORCING_MODE = standard` dan `AWS_VPC_K8S_CNI_EXTERNALSNAT = false`, ketika Pod Anda berkomunikasi dengan titik akhir di luar VPC Anda, grup keamanan node akan digunakan, bukan grup keamanan apa pun yang telah Anda tetapkan ke Pod Anda.

Jika Anda juga menggunakan [grup keamanan untuk Pod](#), dengan `POD_SECURITY_GROUP_ENFORCING_MODE = strict`, saat Pods berkomunikasi dengan titik akhir di luar VPC, grup keamanan Pod 's akan digunakan.

Tingkatkan alamat IP yang tersedia untuk node Amazon EKS Anda

Anda dapat meningkatkan jumlah alamat IP yang dapat ditetapkan node ke Pod dengan menetapkan awalan IP, daripada menetapkan alamat IP sekunder individual ke node Anda.

Prasyarat

- Anda membutuhkan cluster yang ada. Untuk menyebarkan satu, lihat [the section called "Membuat kluster"](#).
- Subnet tempat node Amazon EKS Anda berada harus memiliki blok yang cukup berdekatan /28 (untuk IPv4 cluster) atau (untuk cluster) Classless Inter-Domain IPv6 Routing /80 (CIDR). Anda hanya dapat memiliki node Linux dalam sebuah IPv6 cluster. Menggunakan awalan IP dapat gagal jika alamat IP tersebar di seluruh subnet CIDR. Sebaiknya lakukan hal berikut:
 - Menggunakan reservasi CIDR subnet sehingga meskipun ada alamat IP dalam rentang cadangan masih digunakan, setelah dirilis, alamat IP tidak dipindahkan. Ini memastikan bahwa awalan tersedia untuk alokasi tanpa segmentasi.
 - Gunakan subnet baru yang secara khusus digunakan untuk menjalankan beban kerja yang awalan IP ditetapkan. Beban kerja Windows dan Linux dapat berjalan di subnet yang sama saat menetapkan awalan IP.
- Untuk menetapkan awalan IP ke node Anda, node Anda harus berbasis Nitro. AWS Instans yang tidak berbasis Nitro terus mengalokasikan alamat IP sekunder individual, tetapi memiliki jumlah alamat IP yang jauh lebih rendah untuk ditetapkan ke Pod daripada instans berbasis Nitro.
- Untuk cluster dengan node Linux saja - Jika cluster Anda dikonfigurasi untuk IPv4 keluarga, Anda harus memiliki versi 1.9.0 atau yang lebih baru dari plugin Amazon VPC CNI untuk add-on Kubernetes diinstal. Anda dapat memeriksa versi Anda saat ini dengan perintah berikut.


```
kubectl describe daemonset aws-node --namespace kube-system | grep Image | cut -d "/"
-f 2
```

Jika cluster Anda dikonfigurasi untuk IPv6 keluarga, Anda harus menginstal 1.10.1 versi add-on. Jika versi plugin Anda lebih awal dari versi yang diperlukan, Anda harus memperbaruinya. Untuk informasi selengkapnya, lihat bagian update [Assign IPs to Pod dengan Amazon VPC CNI](#).

- Untuk cluster dengan node Windows saja
 - Cluster Anda dan versi platformnya harus di, atau lebih lambat dari versi dalam tabel berikut. Untuk memutakhirkan versi cluster Anda, lihat [the section called “Perbarui versi Kubernetes”](#). Jika klaster Anda tidak pada versi platform minimum, maka Anda tidak dapat menetapkan awalan IP ke node Anda hingga Amazon EKS memperbarui versi platform Anda.

Versi Kubernetes	Versi platform
1.27	eks.3
1.26	eks.4

Anda dapat memeriksa Kubernetes dan versi platform Anda saat ini dengan mengganti *my-cluster* perintah berikut dengan nama cluster Anda dan kemudian menjalankan perintah yang dimodifikasi: `aws eks describe-cluster --name my-cluster --query 'cluster.{ "Kubernetes Version": version, "Platform Version": platformVersion}'`

- Anda harus mengaktifkan dukungan Windows untuk cluster Anda. Untuk informasi selengkapnya, lihat [the section called “Aktifkan dukungan Windows”](#).

Tetapkan awalan alamat IP ke node

Konfigurasi cluster Anda untuk menetapkan awalan alamat IP ke node. Selesaikan prosedur yang cocok dengan sistem operasi node Anda.

Linux

1. Aktifkan parameter untuk menetapkan awalan ke antarmuka jaringan untuk Amazon VPC CNI. DaemonSet Saat Anda menerapkan klaster, versi 1.10.1 atau yang lebih baru dari plugin Amazon VPC CNI untuk add-on Kubernetes akan di-deploy dengannya. Jika Anda membuat

cluster dengan IPv6 keluarga, pengaturan ini disetel ke secara `true` default. Jika Anda membuat cluster dengan IPv4 keluarga, pengaturan ini disetel ke secara `false` default.

```
kubectl set env daemonset aws-node -n kube-system ENABLE_PREFIX_DELEGATION=true
```

Important

Bahkan jika subnet Anda memiliki alamat IP yang tersedia, jika subnet tidak memiliki /28 blok yang berdekatan, Anda akan melihat kesalahan berikut di plugin Amazon VPC CNI untuk log Kubernetes.

```
InsufficientCidrBlocks: The specified subnet does not have enough free cidr blocks to satisfy the request
```

Hal ini dapat terjadi karena fragmentasi alamat IP sekunder yang ada tersebar di seluruh subnet. Untuk mengatasi kesalahan ini, buat subnet baru dan luncurkan Pod di sana, atau gunakan reservasi CIDR EC2 subnet Amazon untuk memesan ruang di dalam subnet untuk digunakan dengan penetapan awalan. Untuk informasi selengkapnya, lihat [Reservasi CIDR Subnet](#) di Panduan Pengguna Amazon VPC.

2. Jika Anda berencana untuk menerapkan grup node terkelola tanpa template peluncuran, atau dengan template peluncuran yang belum Anda tentukan ID AMI, dan Anda menggunakan versi plugin Amazon VPC CNI untuk Kubernetes pada atau lebih baru dari versi yang tercantum dalam prasyarat, lalu lewati ke langkah berikutnya. Grup node terkelola secara otomatis menghitung jumlah maksimum Pod untuk Anda.

Jika Anda menerapkan grup node yang dikelola sendiri atau grup node terkelola dengan template peluncuran yang telah Anda tentukan ID AMI, Anda harus menentukan jumlah maksimum Pod yang direkomendasikan Amazon EKS untuk node Anda. Ikuti petunjuk di [Amazon EKS Pod maksimum yang direkomendasikan untuk setiap jenis EC2 instans Amazon](#), tambahkan `--cni-prefix-delegation-enabled` ke langkah 3. Perhatikan output untuk digunakan di langkah selanjutnya.

Important

Grup node terkelola memberlakukan jumlah maksimum pada nilai `maxPods` Untuk contoh dengan kurang dari 30 v CPUs jumlah maksimum adalah 110 dan untuk semua contoh

lainnya jumlah maksimum adalah 250. Jumlah maksimum ini diterapkan apakah delegasi awalan diaktifkan atau tidak.

3. Jika Anda menggunakan cluster yang dikonfigurasi IPv6, lewati ke langkah berikutnya.

Tentukan parameter di salah satu opsi berikut. Untuk menentukan opsi mana yang tepat untuk Anda dan nilai apa yang akan diberikan untuknya, lihat [WARM_PREFIX_TARGET](#), [WARM_IP_TARGET](#), dan [MINIMUM_IP_TARGET](#) aktif. GitHub

Anda dapat mengganti *example values* dengan nilai yang lebih besar dari nol.

- WARM_PREFIX_TARGET

```
kubectl set env ds aws-node -n kube-system WARM_PREFIX_TARGET=1
```

- WARM_IP_TARGET atau MINIMUM_IP_TARGET — Jika salah satu nilai ditetapkan, itu akan menggantikan nilai yang ditetapkan untuk WARM_PREFIX_TARGET

```
kubectl set env ds aws-node -n kube-system WARM_IP_TARGET=5
```

```
kubectl set env ds aws-node -n kube-system MINIMUM_IP_TARGET=2
```

4. Buat salah satu jenis grup node berikut dengan setidaknya satu jenis instans Amazon EC2 Nitro Amazon Linux 2. Untuk daftar jenis instans Nitro, lihat [Instans yang dibangun di Sistem Nitro di Panduan Pengguna Amazon EC2](#). Kemampuan ini tidak didukung pada Windows. Untuk opsi yang disertakan **110**, ganti dengan nilai dari langkah 3 (disarankan), atau nilai Anda sendiri.

- Dikelola sendiri — Menerapkan grup node menggunakan instruksi di [Buat node Amazon Linux yang dikelola](#) sendiri. Tentukan teks berikut untuk BootstrapArguments parameter.

```
--use-max-pods false --kubelet-extra-args '--max-pods=110'
```

Jika Anda menggunakan eksctl untuk membuat grup node, Anda dapat menggunakan perintah berikut.

```
eksctl create nodegroup --cluster my-cluster --managed=false --max-pods-per-node 110
```

- Dikelola - Terapkan grup node Anda menggunakan salah satu opsi berikut:

- Tanpa template peluncuran atau dengan template peluncuran tanpa ID AMI yang ditentukan — Selesaikan prosedur di [Buat grup node terkelola untuk klaster Anda](#). Grup node terkelola secara otomatis menghitung max-pods nilai yang direkomendasikan Amazon EKS untuk Anda.
- Dengan template peluncuran dengan ID AMI tertentu — Dalam template peluncuran Anda, tentukan ID AMI Amazon EKS yang dioptimalkan, atau AMI kustom yang dibuat dari AMI Amazon EKS yang dioptimalkan, lalu [terapkan grup node menggunakan template peluncuran](#) dan berikan data pengguna berikut di template peluncuran. Data pengguna ini meneruskan argumen ke dalam bootstrap.sh file. Untuk informasi selengkapnya tentang file bootstrap, lihat [bootstrap.sh](#) di GitHub.

```
/etc/eks/bootstrap.sh my-cluster \  
--use-max-pods false \  
--kubelet-extra-args '--max-pods=110'
```

Jika Anda menggunakan eksctl untuk membuat grup node, Anda dapat menggunakan perintah berikut.

```
eksctl create nodegroup --cluster my-cluster --max-pods-per-node 110
```

Jika Anda telah membuat AMI kustom yang tidak dibangun dari AMI Amazon EKS yang dioptimalkan, maka Anda perlu membuat konfigurasi sendiri.

Note

Jika Anda juga ingin menetapkan alamat IP ke Pod dari subnet yang berbeda dari instans, maka Anda perlu mengaktifkan kapabilitas pada langkah ini. Untuk informasi selengkapnya, lihat [the section called “Jaringan kustom”](#).

Windows

1. Aktifkan penugasan awalan IP.
 - a. Buka amazon-vpc-cni ConfigMap untuk mengedit.

```
kubectl edit configmap -n kube-system amazon-vpc-cni -o yaml
```

- b. Tambahkan baris berikut ke data bagian.

```
enable-windows-prefix-delegation: "true"
```

- c. Simpan file dan tutup editor.
- d. Konfirmasikan bahwa baris telah ditambahkan ke ConfigMap.

```
kubectl get configmap -n kube-system amazon-vpc-cni -o "jsonpath={.data.enable-windows-prefix-delegation}"
```

Jika output yang dikembalikan tidak `true`, maka mungkin ada kesalahan. Coba selesaikan langkahnya lagi.

Important

Bahkan jika subnet Anda memiliki alamat IP yang tersedia, jika subnet tidak memiliki /28 blok yang berdekatan, Anda akan melihat kesalahan berikut di plugin Amazon VPC CNI untuk log Kubernetes.

```
InsufficientCidrBlocks: The specified subnet does not have enough free cidr blocks to satisfy the request
```

Hal ini dapat terjadi karena fragmentasi alamat IP sekunder yang ada tersebar di seluruh subnet. Untuk mengatasi kesalahan ini, buat subnet baru dan luncurkan Pod di sana, atau gunakan reservasi CIDR EC2 subnet Amazon untuk memesan ruang di dalam subnet untuk digunakan dengan penetapan awalan. Untuk informasi selengkapnya, lihat [Reservasi CIDR Subnet](#) di Panduan Pengguna Amazon VPC.

2. (Opsional) Tentukan konfigurasi tambahan untuk mengontrol perilaku pra-penskalaan dan penskalaan dinamis untuk kluster Anda. Untuk informasi selengkapnya, lihat [Opsional konfigurasi dengan mode Delegasi Awalan di Windows](#) aktif. GitHub

- a. Buka `amazon-vpc-cni` ConfigMap untuk mengedit.

```
kubectl edit configmap -n kube-system amazon-vpc-cni -o yaml
```

- b. Ganti *example values* dengan nilai lebih besar dari nol dan tambahkan entri yang Anda butuhkan ke data bagian ConfigMap. Jika Anda menetapkan nilai untuk salah satu `warm-ip-`

target atau minimum-ip-target, nilai akan menggantikan nilai apa pun yang ditetapkan untuk warm-prefix-target

```
warm-prefix-target: "1"
warm-ip-target: "5"
minimum-ip-target: "2"
```

c. Simpan file dan tutup editor.

3. Buat grup node Windows dengan setidaknya satu jenis instans Amazon EC2 Nitro. Untuk daftar jenis instans Nitro, lihat [Instans yang dibangun di Sistem Nitro di Panduan Pengguna Amazon EC2](#). Secara default, jumlah maksimum Pod yang dapat Anda deploy ke sebuah node adalah 110. Jika Anda ingin menambah atau mengurangi angka itu, tentukan yang berikut ini dalam data pengguna untuk konfigurasi bootstrap. Ganti *max-pods-quantity* dengan nilai pod maks Anda.

```
-KubeletExtraArgs '--max-pods=max-pods-quantity'
```

Jika Anda menerapkan grup node terkelola, konfigurasi ini perlu ditambahkan dalam template peluncuran. Untuk informasi selengkapnya, lihat [the section called “Templat peluncuran”](#). Untuk informasi selengkapnya tentang parameter konfigurasi untuk skrip bootstrap Windows, lihat [the section called “Parameter konfigurasi skrip bootstrap”](#).

Tentukan Pod maksimal dan alamat IP yang tersedia

1. Setelah node Anda di-deploy, lihat node di cluster Anda.

```
kubectl get nodes
```

Contoh output adalah sebagai berikut.

NAME	STATUS	ROLES	AGE	VERSION
ip-192-168-22-103.region-code.compute.internal eks-6b7464	Ready	<none>	19m	v1.XX.X-
ip-192-168-97-94.region-code.compute.internal eks-6b7464	Ready	<none>	19m	v1.XX.X-

2. Jelaskan salah satu node untuk menentukan nilai max-pods untuk node dan jumlah alamat IP yang tersedia. Ganti *192.168.30.193* dengan IPv4 alamat atas nama salah satu node Anda yang dikembalikan pada output sebelumnya.

```
kubectl describe node ip-192-168-30-193.region-code.compute.internal | grep 'pods\|PrivateIPv4Address'
```

Contoh output adalah sebagai berikut.

```
pods: 110
vpc.amazonaws.com/PrivateIPv4Address: 144
```

Pada output sebelumnya, 110 adalah jumlah maksimum Pod yang Kubernetes akan deploy ke node, meskipun 144 alamat IP tersedia.

Menetapkan grup keamanan ke Pod individual

Berlaku untuk: Node Linux dengan EC2 instans Amazon

Berlaku untuk: Subnet pribadi

Grup keamanan untuk Pod mengintegrasikan grup EC2 keamanan Amazon dengan Kubernetes Pods. Anda dapat menggunakan grup EC2 keamanan Amazon untuk menentukan aturan yang memungkinkan lalu lintas jaringan masuk dan keluar ke dan dari Pod yang Anda terapkan ke node yang berjalan di banyak jenis EC2 instans Amazon dan Fargate. Untuk penjelasan rinci tentang kemampuan ini, lihat posting blog [Introducing security groups for Pods](#).

Kompatibilitas dengan plugin Amazon VPC CNI untuk fitur Kubernetes

Anda dapat menggunakan grup keamanan untuk Pod dengan fitur berikut:

- IPv4 Terjemahan Alamat Jaringan Sumber - Untuk informasi lebih lanjut, lihat [the section called “Lalu lintas keluar”](#).
- IPv6 alamat ke klaster, Pod, dan layanan - Untuk informasi selengkapnya, lihat [the section called “IPv6”](#).
- Membatasi lalu lintas menggunakan kebijakan jaringan Kubernetes - Untuk informasi selengkapnya, lihat [the section called “Kebijakan Kubernetes”](#)

Pertimbangan

Sebelum menerapkan grup keamanan untuk Pod, pertimbangkan batasan dan ketentuan berikut:

- Grup keamanan untuk Pod tidak dapat digunakan dengan node Windows.
- Grup keamanan untuk Pod dapat digunakan dengan cluster yang dikonfigurasi untuk IPv6 keluarga yang berisi EC2 node Amazon dengan menggunakan plugin Amazon VPC CNI versi 1.16.0 atau yang lebih baru. Anda dapat menggunakan grup keamanan untuk Pod dengan cluster yang mengonfigurasi IPv6 keluarga yang hanya berisi node Fargate dengan menggunakan plugin Amazon VPC CNI versi 1.7.7 atau yang lebih baru. Untuk informasi selengkapnya, lihat [the section called “IPv6”](#)
- Grup keamanan untuk Pod didukung oleh sebagian besar keluarga EC2 instans Amazon [berbasis Nitro](#), meskipun tidak oleh semua generasi keluarga. Misalnya, keluarga dan generasi m5 c5r5, m6gc6g,,, dan r6g contoh didukung. Tidak ada jenis instance dalam t keluarga yang didukung. Untuk daftar lengkap jenis instans yang didukung, lihat file [limits.go](#) di. GitHub Node Anda harus menjadi salah satu jenis instance terdaftar yang ada `IsTrunkingCompatible: true` di file itu.
- Jika Anda menggunakan jaringan kustom dan grup keamanan untuk Pod bersama-sama, grup keamanan yang ditentukan oleh grup keamanan untuk Pod akan digunakan sebagai ganti grup keamanan yang ditentukan dalam `ENIConfig`.
- Jika Anda menggunakan versi 1.10.2 atau versi sebelumnya dari plugin Amazon VPC CNI dan Anda menyertakan `terminationGracePeriodSeconds` setelan dalam spesifikasi Pod Anda, nilai untuk setelan tidak bisa nol.
- Jika Anda menggunakan versi 1.10 atau sebelumnya dari plugin Amazon VPC CNI, atau versi 1.11 dengan `POD_SECURITY_GROUP_ENFORCING_MODE =`, yang merupakan setelan default `strict`, maka jenis layanan Kubernetes `NodePort` dan `LoadBalancer` menggunakan target instance dengan `externalTrafficPolicy` set to `Local` tidak didukung dengan Pod yang Anda tetapkan ke grup keamanan. Untuk informasi selengkapnya tentang cara menggunakan penyeimbang beban dengan target instans, lihat [the section called “Menyeimbangkan beban jaringan”](#).
- Jika Anda menggunakan versi 1.10 atau versi sebelumnya dari plugin Amazon VPC CNI atau versi 1.11 dengan `POD_SECURITY_GROUP_ENFORCING_MODE =strict`, yang merupakan pengaturan default, NAT sumber dinonaktifkan untuk lalu lintas keluar dari Pod dengan grup keamanan yang ditetapkan sehingga aturan grup keamanan keluar diterapkan. Untuk mengakses internet, Pod dengan grup keamanan yang ditetapkan harus diluncurkan pada node yang digunakan dalam subnet pribadi yang dikonfigurasi dengan gateway atau instance NAT. Pod dengan grup keamanan yang ditugaskan dan di-deploy ke subnet publik tidak dapat mengakses internet.

Jika Anda menggunakan versi 1.11 atau versi yang lebih baru dari plugin dengan `POD_SECURITY_GROUP_ENFORCING_MODE=standard`, lalu lintas Pod yang ditujukan untuk di luar VPC diterjemahkan ke alamat IP antarmuka jaringan utama instans. Untuk lalu lintas ini, aturan dalam grup keamanan untuk antarmuka jaringan utama digunakan, bukan aturan dalam grup keamanan Pod.

- Untuk menggunakan kebijakan jaringan Calico dengan Pod yang memiliki grup keamanan terkait, Anda harus menggunakan versi 1.11.0 atau yang lebih baru dari plugin Amazon VPC CNI dan set `POD_SECURITY_GROUP_ENFORCING_MODE=standard`. Jika tidak, arus lalu lintas ke dan dari Pod dengan grup keamanan terkait tidak tunduk pada penegakan kebijakan jaringan Calico dan terbatas pada penegakan kelompok EC2 keamanan Amazon saja. Untuk memperbarui versi CNI VPC Amazon Anda, lihat [the section called “Amazon VPC CNI”](#)
- Pod yang berjalan di EC2 node Amazon yang menggunakan grup keamanan dalam kluster yang [NodeLocal DNSCache](#) digunakan hanya didukung dengan versi 1.11.0 atau yang lebih baru dari plugin Amazon VPC CNI dan dengan `POD_SECURITY_GROUP_ENFORCING_MODE=standard`. Untuk memperbarui versi plugin Amazon VPC CNI Anda, lihat [the section called “Amazon VPC CNI”](#)
- Grup keamanan untuk Pod dapat menyebabkan latensi startup Pod yang lebih tinggi untuk Pod dengan churn tinggi. Hal ini disebabkan pembatasan tingkat dalam pengontrol sumber daya.
- Cakupan grup EC2 keamanan berada di tingkat POD - Untuk informasi selengkapnya, lihat [Grup keamanan](#).

Jika Anda menetapkan `POD_SECURITY_GROUP_ENFORCING_MODE=standard` dan `AWS_VPC_K8S_CNI_EXTERNALSNAT=false`, lalu lintas yang ditujukan untuk titik akhir di luar VPC menggunakan grup keamanan node, bukan grup keamanan Pod.

Konfigurasi plugin Amazon VPC CNI untuk Kubernetes untuk grup keamanan untuk Pod Amazon EKS

Jika Anda menggunakan Pod dengan EC2 instans Amazon, Anda perlu mengonfigurasi plugin Amazon VPC CNI untuk Kubernetes untuk grup keamanan

Jika Anda hanya menggunakan Pod Fargate, dan tidak memiliki EC2 node Amazon di kluster Anda, lihat [the section called “SecurityGroupPolicy”](#)

1. Periksa plugin Amazon VPC CNI Anda saat ini untuk versi Kubernetes dengan perintah berikut:

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |
cut -d : -f 3
```

Contoh output adalah sebagai berikut.

```
v1.7.6
```

Jika plugin Amazon VPC CNI Anda untuk versi Kubernetes lebih awal dari 1.7.7, maka perbarui plugin ke versi atau yang lebih baru. 1.7.7 Untuk informasi selengkapnya, lihat [the section called “Amazon VPC CNI”](#)

2. Tambahkan tautan:iam/home#/policies/arn:aws:iam::AWS:policy/AmazonEKSVPCResourceController [Pengontrol Amazon, type="console "] kebijakan IAM terkelola ke [peran kluster yang terkait dengan kluster EKSVPCResource](#) Amazon EKS Anda. Kebijakan ini memungkinkan peran untuk mengelola antarmuka jaringan, alamat IP pribadi mereka, dan lampiran serta detasemen mereka ke dan dari instance jaringan.

- a. Ambil nama peran IAM cluster Anda dan simpan dalam variabel. Ganti *my-cluster* dengan nama kluster Anda.

```
cluster_role=$(aws eks describe-cluster --name my-cluster --query cluster.roleArn
--output text | cut -d / -f 2)
```

- b. Lampirkan kebijakan pada peran tersebut.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSVPCResourceController --role-name $cluster_role
```

3. Aktifkan add-on Amazon VPC CNI untuk mengelola antarmuka jaringan untuk Pod dengan menyetel variabel ke dalam `ENABLE_POD_ENI`. `true` `aws-node` DaemonSet Setelah pengaturan ini diatur ke `true`, untuk setiap node di cluster add-on membuat sumber daya `cninode` khusus. Pengendali sumber daya VPC membuat dan melampirkan satu antarmuka jaringan khusus yang disebut antarmuka jaringan trunk dengan deskripsi `aws-k8s-trunk-eni`.

```
kubectl set env daemonset aws-node -n kube-system ENABLE_POD_ENI=true
```

Note

Antarmuka jaringan trunk termasuk dalam jumlah maksimum antarmuka jaringan yang didukung oleh tipe instans. Untuk daftar jumlah maksimum antarmuka jaringan yang didukung oleh setiap jenis instans, lihat [alamat IP per antarmuka jaringan per jenis instans](#) di Panduan EC2 Pengguna Amazon. Jika simpul Anda sudah memiliki jumlah maksimum antarmuka jaringan standar yang terlampir padanya, maka pengendali sumber daya VPC akan menyimpan sebuah ruang. Anda harus menurunkan skala Pod yang sedang berjalan agar pengontrol dapat melepaskan dan menghapus antarmuka jaringan standar, membuat antarmuka jaringan trunk, dan melampirkannya ke instance.

4. Anda dapat melihat node mana yang memiliki sumber daya CNINode khusus dengan perintah berikut. Jika `No resources found` dikembalikan, maka tunggu beberapa detik dan coba lagi. Langkah sebelumnya memerlukan restart plugin Amazon VPC CNI untuk Kubernetes Pods, yang membutuhkan waktu beberapa detik.

```
kubectl get cninode -A
NAME FEATURES
ip-192-168-64-141.us-west-2.compute.internal [{"name":"SecurityGroupsForPods"}]
ip-192-168-7-203.us-west-2.compute.internal [{"name":"SecurityGroupsForPods"}]
```

Jika Anda menggunakan versi VPC CNI yang lebih lama dari 1.15, label node digunakan sebagai pengganti sumber daya khusus. CNINode Anda dapat melihat node mana yang memiliki label `node aws-k8s-trunk-eni` diatur `true` dengan perintah berikut. Jika `No resources found` dikembalikan, maka tunggu beberapa detik dan coba lagi. Langkah sebelumnya memerlukan restart plugin Amazon VPC CNI untuk Kubernetes Pods, yang membutuhkan waktu beberapa detik.

```
kubectl get nodes -o wide -l vpc.amazonaws.com/has-trunk-attached=true
-
```

Setelah antarmuka jaringan trunk dibuat, Pod diberi alamat IP sekunder dari trunk atau antarmuka jaringan standar. Antarmuka trunk secara otomatis dihapus jika simpul dihapus.

Saat Anda menerapkan grup keamanan untuk Pod di langkah selanjutnya, pengontrol sumber daya VPC membuat antarmuka jaringan khusus yang disebut antarmuka jaringan cabang dengan deskripsi `aws-k8s-branch-eni` dan mengaitkan grup keamanan ke Pod tersebut. Antarmuka

jaringan cabang dibuat sebagai tambahan dari antarmuka jaringan standar dan trunk yang terlampir pada simpul.

Jika Anda menggunakan probe keaktifan atau kesiapan, maka Anda juga perlu menonaktifkan demux awal TCP, sehingga kubelet dapat terhubung ke Pod pada antarmuka jaringan cabang menggunakan TCP. Untuk menonaktifkan TCP demux awal, jalankan perintah berikut:

```
kubectl patch daemonset aws-node -n kube-system \
  -p '{"spec": {"template": {"spec": {"initContainers": [{"env":
[{"name": "DISABLE_TCP_EARLY_DEMUX", "value": "true"}], "name": "aws-vpc-cni-init"}]}}}'
```

Note

Jika Anda menggunakan 1.11.0 atau yang lebih baru dari plugin Amazon VPC CNI untuk Kubernetes add-on dan set `POD_SECURITY_GROUP_ENFORCING_MODE=standard`, seperti yang dijelaskan pada langkah berikutnya, maka Anda tidak perlu menjalankan perintah sebelumnya.

5. Jika klaster Anda menggunakan `NodeLocal DNSCache`, atau Anda ingin menggunakan kebijakan jaringan Calico dengan Pod yang memiliki grup keamanan sendiri, atau Anda memiliki jenis layanan Kubernetes `NodePort` dan `LoadBalancer` menggunakan target instans dengan `externalTrafficPolicy` set ke Pod yang ingin Anda tetapkan ke `Local` grup keamanan, maka Anda harus menggunakan 1.11.0 versi atau yang lebih baru dari plugin Amazon VPC CNI untuk add-on Kubernetes, dan Anda harus mengaktifkan pengaturan berikut:

```
kubectl set env daemonset aws-node -n kube-system
  POD_SECURITY_GROUP_ENFORCING_MODE=standard
```

PENTING: Aturan grup keamanan Pod tidak diterapkan pada lalu lintas antar Pod atau antara Pod dan layanan, seperti **kubelet** atau **nodeLocalDNS**, yang berada di node yang sama. Pod yang menggunakan grup keamanan berbeda pada node yang sama tidak dapat berkomunikasi karena mereka dikonfigurasi dalam subnet yang berbeda, dan routing dinonaktifkan di antara subnet ini. Lalu lintas keluar dari Pod ke alamat di luar VPC adalah alamat jaringan yang diterjemahkan ke alamat IP antarmuka jaringan utama instans (kecuali jika Anda juga `AWS_VPC_K8S_CNI_EXTERNALSNAT=true` telah menyetelnya). Untuk lalu lintas ini, aturan dalam grup keamanan untuk antarmuka jaringan utama digunakan, bukan aturan dalam grup keamanan

Pod. ** Agar pengaturan ini diterapkan pada Pod yang ada, Anda harus me-restart Pod atau node tempat Pod berjalan.

6. Untuk melihat cara menggunakan kebijakan grup keamanan untuk Pod Anda, lihat [the section called “SecurityGroupPolicy”](#).

Menggunakan kebijakan grup keamanan untuk Amazon EKS Pod

Untuk menggunakan grup keamanan untuk Pod, Anda harus memiliki grup keamanan yang sudah ada. Langkah-langkah berikut menunjukkan cara menggunakan kebijakan grup keamanan untuk sebuah Pod. Kecuali dinyatakan lain, selesaikan semua langkah dari terminal yang sama karena variabel digunakan dalam langkah-langkah berikut yang tidak bertahan di seluruh terminal.

Jika Anda memiliki Pod dengan EC2 instans Amazon, Anda harus mengonfigurasi plugin sebelum menggunakan prosedur ini. Untuk informasi selengkapnya, lihat [the section called “Konfigurasi”](#).

1. Buat namespace Kubernetes untuk menyebarkan sumber daya. Anda dapat mengganti *my-namespace* dengan nama namespace yang ingin Anda gunakan.

```
kubectl create namespace my-namespace
```

2. Menerapkan Amazon EKS SecurityGroupPolicy ke cluster Anda.
 - a. Salin konten berikut ke perangkat Anda. Anda dapat mengganti *podSelector* dengan *serviceAccountSelector* jika Anda lebih suka memilih Pod berdasarkan label akun layanan. Anda harus menentukan satu selektor atau yang lainnya. Sebuah kosong *podSelector* (contoh: `podSelector: {}`) memilih semua Pod di namespace. Anda dapat *my-role* mengubah nama peran Anda. *serviceAccountSelector* yang kosong memilih semua akun layanan di namespace. Anda dapat mengganti *my-security-group-policy* dengan nama untuk Anda SecurityGroupPolicy dan *my-namespace* dengan namespace yang ingin Anda buat. SecurityGroupPolicy

Anda harus mengganti *my_pod_security_group_id* dengan ID grup keamanan yang ada. Jika Anda tidak memiliki grup keamanan yang ada, maka Anda harus membuatnya. Untuk informasi selengkapnya, lihat [Grup EC2 keamanan Amazon untuk instans Linux](#) di [Panduan EC2 Pengguna Amazon](#). Anda dapat menentukan 1-5 grup IDs keamanan. Jika Anda menentukan lebih dari satu ID, maka kombinasi semua aturan di semua grup keamanan akan efektif untuk Pod yang dipilih.

```
cat >my-security-group-policy.yaml <<EOF
```

```
apiVersion: vpcresources.k8s.aws/v1beta1
kind: SecurityGroupPolicy
metadata:
  name: my-security-group-policy
  namespace: my-namespace
spec:
  podSelector:
    matchLabels:
      role: my-role
  securityGroups:
    groupIds:
      - my_pod_security_group_id
EOF
```

Important

Grup atau grup keamanan yang Anda tentukan untuk Pod Anda harus memenuhi kriteria berikut:

- Mereka harus ada. Jika mereka tidak ada, maka, ketika Anda menerapkan Pod yang cocok dengan pemilih, Pod Anda tetap terjebak dalam proses pembuatan. Jika Anda mendeskripsikan Pod, Anda akan melihat pesan galat yang mirip dengan yang berikut: An error occurred (InvalidSecurityGroupID.NotFound) when calling the CreateNetworkInterface operation: The securityGroup ID '*sg-05b1d815d1EXAMPLE*' does not exist
- Mereka harus mengizinkan komunikasi masuk dari grup keamanan yang diterapkan ke node Anda (untuk kubelet) melalui port apa pun yang telah Anda konfigurasi probe.
- Mereka harus mengizinkan komunikasi keluar TCP dan UDP port 53 ke grup keamanan yang ditetapkan ke Pod (atau node tempat Pod berjalan) yang menjalankan CoreDNS. Grup keamanan untuk Pod CoreDNS Anda harus mengizinkan lalu lintas TCP masuk UDP dan port 53 dari grup keamanan yang Anda tentukan.
- Mereka harus memiliki aturan inbound dan outbound yang diperlukan untuk berkomunikasi dengan Pod lain yang perlu mereka komunikasikan.
- Mereka harus memiliki aturan yang memungkinkan Pod untuk berkomunikasi dengan control plane Kubernetes jika Anda menggunakan grup keamanan dengan Fargate. Cara termudah untuk melakukannya adalah dengan menentukan grup keamanan kluster sebagai salah satu grup keamanan.

Kebijakan grup keamanan hanya berlaku untuk Pod yang baru dijadwalkan. Mereka tidak memengaruhi menjalankan Pod.

b. Deploy kebijakan.

```
kubectl apply -f my-security-group-policy.yaml
```

3. Menerapkan aplikasi sampel dengan label yang cocok dengan *my-role* nilai *podSelector* yang Anda tentukan pada langkah sebelumnya.

- a. Salin konten berikut ke perangkat Anda. Ganti *example values* dengan milik Anda sendiri dan kemudian jalankan perintah yang dimodifikasi. Jika Anda mengganti *my-role*, pastikan itu sama dengan nilai yang Anda tentukan untuk pemilih pada langkah sebelumnya.

```
cat >sample-application.yaml <<EOF
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-deployment
  namespace: my-namespace
  labels:
    app: my-app
spec:
  replicas: 4
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
        role: my-role
    spec:
      terminationGracePeriodSeconds: 120
      containers:
        - name: nginx
          image: public.ecr.aws/nginx/nginx:1.23
          ports:
            - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
```

```

name: my-app
namespace: my-namespace
labels:
  app: my-app
spec:
  selector:
    app: my-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
EOF

```

- b. Deploy aplikasi dengan perintah berikut. Ketika Anda menerapkan aplikasi, plugin Amazon VPC CNi untuk Kubernetes cocok dengan role label dan grup keamanan yang Anda tentukan pada langkah sebelumnya diterapkan ke Pod.

```
kubectl apply -f sample-application.yaml
```

4. Lihat Pod yang di-deploy dengan aplikasi sampel. Untuk sisa topik ini, terminal ini disebut sebagai Terminal A.

```
kubectl get pods -n my-namespace -o wide
```

Contoh output adalah sebagai berikut.

NAME	READY	STATUS	RESTARTS	AGE	IP
NODE			NOMINATED	NODE	READINESS GATES
my-deployment-5df6f7687b-4fbjm	1/1	Running	0	7m51s	192.168.53.48
ip-192-168-33-28.region-code.compute.internal			<none>		<none>
my-deployment-5df6f7687b-j9fl4	1/1	Running	0	7m51s	192.168.70.145
ip-192-168-92-33.region-code.compute.internal			<none>		<none>
my-deployment-5df6f7687b-rjxcz	1/1	Running	0	7m51s	192.168.73.207
ip-192-168-92-33.region-code.compute.internal			<none>		<none>
my-deployment-5df6f7687b-zmb42	1/1	Running	0	7m51s	192.168.63.27
ip-192-168-33-28.region-code.compute.internal			<none>		<none>

Note

Coba tips ini jika ada Pod yang macet.

- Jika ada Pod yang macet dalam Waiting status, maka jalankan `kubectl describe pod my-deployment-xxxxxxxxxx-xxxxx -n my-namespace`. Jika Anda melihatnya `Insufficient permissions: Unable to create Elastic Network Interface.`, konfirmasikan bahwa Anda menambahkan kebijakan IAM ke peran kluster IAM di langkah sebelumnya.
- Jika ada Pod yang terjebak dalam Pending status, konfirmasikan bahwa jenis instance node Anda terdaftar di [limits.go](https://docs.aws.amazon.com/eks/latest/userguide/limits.html) dan bahwa produk dari jumlah maksimum antarmuka jaringan cabang yang didukung oleh tipe instance dikalikan kali jumlah node dalam grup node Anda belum terpenuhi. Misalnya, sebuah instans `m5.large` mendukung sembilan antarmuka jaringan cabang. Jika grup simpul Anda memiliki lima simpul, maka maksimum 45 antarmuka jaringan cabang dapat dibuat untuk grup simpul. Pod ke-46 yang Anda coba gunakan akan berada dalam Pending status sampai Pod lain yang memiliki grup keamanan terkait dihapus.

Jika Anda menjalankan `kubectl describe pod my-deployment-xxxxxxxxxx-xxxxx -n my-namespace` dan melihat pesan yang serupa dengan pesan berikut, maka dapat diabaikan dengan aman. Pesan ini mungkin muncul ketika plugin Amazon VPC CNI untuk Kubernetes mencoba mengatur jaringan host dan gagal saat antarmuka jaringan sedang dibuat. Plugin mencatat peristiwa ini sampai antarmuka jaringan dibuat.

```
Failed to create Pod sandbox: rpc error: code = Unknown desc = failed to set up
sandbox container "e24268322e55c8185721f52df6493684f6c2c3bf4fd59c9c121fd4cdc894579f"
network for Pod "my-deployment-5df6f7687b-4fbjm": networkPlugin
cni failed to set up Pod "my-deployment-5df6f7687b-4fbjm-c89wx_my-namespace" network:
add cmd: failed to assign an IP address to container
```

Anda tidak dapat melebihi jumlah maksimum Pod yang dapat dijalankan pada tipe instans. Untuk daftar jumlah maksimum Pod yang dapat Anda jalankan pada setiap jenis instance, lihat [eni-max-pods.txt](#) di GitHub. Saat Anda menghapus Pod yang memiliki grup keamanan terkait, atau menghapus node tempat Pod berjalan, pengontrol sumber daya VPC akan menghapus antarmuka jaringan cabang. Jika Anda menghapus kluster dengan Pod menggunakan Pod untuk grup keamanan, maka pengontrol tidak akan menghapus antarmuka jaringan cabang, jadi Anda harus menghapusnya sendiri. Untuk informasi tentang cara menghapus antarmuka jaringan, lihat [Menghapus antarmuka jaringan](#) di Panduan EC2 Pengguna Amazon.

5. Di terminal terpisah, tempelkan ke salah satu Pod. Untuk sisa topik ini, terminal ini disebut sebagai TerminalB. Ganti `5df6f7687b-4fbjm` dengan ID salah satu Pod yang dikembalikan dalam output Anda dari langkah sebelumnya.

```
kubectl exec -it -n my-namespace my-deployment-5df6f7687b-4fbjm -- /bin/bash
```

6. Dari shell masuk TerminalB, konfirmasikan bahwa aplikasi sampel berfungsi.

```
curl my-app
```

Contoh output adalah sebagai berikut.

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
[...]
```

Anda menerima output karena semua Pod yang menjalankan aplikasi dikaitkan dengan grup keamanan yang Anda buat. Grup tersebut berisi aturan yang memungkinkan semua lalu lintas di antara semua Pod yang terkait dengan grup keamanan. Lalu lintas DNS diizinkan keluar dari grup keamanan tersebut ke grup keamanan klaster, yang terkait dengan node Anda. Node menjalankan Pod CoreDNS, yang mana Pod Anda melakukan pencarian nama.

7. Dari TerminalA, hapus aturan grup keamanan yang memungkinkan komunikasi DNS ke grup keamanan klaster dari grup keamanan Anda. Jika Anda tidak menambahkan aturan DNS ke grup keamanan klaster pada langkah sebelumnya, ganti `$my_cluster_security_group_id` dengan ID grup keamanan tempat Anda membuat aturan.

```
aws ec2 revoke-security-group-ingress --group-id $my_cluster_security_group_id --
security-group-rule-ids $my_tcp_rule_id
aws ec2 revoke-security-group-ingress --group-id $my_cluster_security_group_id --
security-group-rule-ids $my_udp_rule_id
```

8. Dari TerminalB, coba akses aplikasi lagi.

```
curl my-app
```

Contoh output adalah sebagai berikut.

```
curl: (6) Could not resolve host: my-app
```

Upaya gagal karena Pod tidak lagi dapat mengakses Pod CoreDNS, yang memiliki grup keamanan klaster yang terkait dengannya. Grup keamanan klaster tidak lagi memiliki aturan grup keamanan yang memungkinkan komunikasi DNS dari grup keamanan yang terkait dengan Pod Anda.

Jika Anda mencoba mengakses aplikasi menggunakan alamat IP yang dikembalikan untuk salah satu Pod pada langkah sebelumnya, Anda masih menerima respons karena semua port diizinkan di antara Pod yang memiliki grup keamanan yang terkait dengannya dan pencarian nama tidak diperlukan.

9. Setelah selesai bereksperimen, Anda dapat menghapus contoh kebijakan grup keamanan, aplikasi, dan grup keamanan yang Anda buat. Jalankan perintah berikut dari TerminalA.

```
kubectl delete namespace my-namespace
aws ec2 revoke-security-group-ingress --group-id $my_pod_security_group_id --
security-group-rule-ids $my_inbound_self_rule_id
wait
sleep 45s
aws ec2 delete-security-group --group-id $my_pod_security_group_id
```

Pasang beberapa antarmuka jaringan ke Pod dengan Multus

Multus CNI adalah plugin Container Network Interface (CNI) untuk Amazon EKS yang memungkinkan melampirkan beberapa antarmuka jaringan ke Pod. Untuk informasi lebih lanjut, lihat dokumentasi [Multus-CNI](#) di GitHub.

Di Amazon EKS, setiap Pod memiliki satu antarmuka jaringan yang ditetapkan oleh plugin Amazon VPC CNI. Dengan Multus, Anda dapat membuat Pod multi-homed yang memiliki beberapa antarmuka. Ini dicapai dengan Multus bertindak sebagai “meta-plugin”; plugin CNI yang dapat memanggil beberapa plugin CNI lainnya. AWS dukungan untuk Multus hadir dikonfigurasi dengan plugin Amazon VPC CNI sebagai plugin delegasi default.

- Amazon EKS tidak akan membangun dan menerbitkan plugin CNI virtualisasi I/O root tunggal (SR-IOV) dan Data Plane Development Kit (DPDK). Namun, Anda dapat mencapai akselerasi paket dengan menghubungkan langsung ke Amazon EC2 Elastic Network Adapters (ENA) melalui perangkat host dan plugin yang dikelola Multus. `ipvlan`

- Amazon EKS mendukung Multus, yang menyediakan proses generik yang memungkinkan rantai sederhana plugin CNI tambahan. Multus dan proses rantai didukung, tetapi tidak AWS akan memberikan dukungan untuk semua plugin CNI yang kompatibel yang dapat dirantai, atau masalah yang mungkin timbul pada plugin CNI yang tidak terkait dengan konfigurasi rantai.
- Amazon EKS menyediakan dukungan dan manajemen siklus hidup untuk plugin Multus, tetapi tidak bertanggung jawab atas alamat IP atau manajemen tambahan yang terkait dengan antarmuka jaringan tambahan. Alamat IP dan pengelolaan antarmuka jaringan default yang menggunakan plugin Amazon VPC CNI tetap tidak berubah.
- Hanya plugin Amazon VPC CNI yang secara resmi didukung sebagai plugin delegasi default. Anda perlu memodifikasi manifes instalasi Multus yang diterbitkan untuk mengkonfigurasi ulang plugin delegasi default ke CNI alternatif jika Anda memilih untuk tidak menggunakan plugin Amazon VPC CNI untuk jaringan utama.
- Multus hanya didukung saat menggunakan Amazon VPC CNI sebagai CNI utama. Kami tidak mendukung Amazon VPC CNI saat digunakan untuk antarmuka tingkat tinggi, sekunder atau lainnya.
- Untuk mencegah plugin Amazon VPC CNI mencoba mengelola antarmuka jaringan tambahan yang ditetapkan ke Pod, tambahkan tag berikut ke antarmuka jaringan:

kunci

```
: node.k8s.amazonaws.com/no_manage
```

nilai

```
: true
```

- Multus kompatibel dengan kebijakan jaringan, tetapi kebijakan tersebut harus diperkaya untuk menyertakan port dan alamat IP yang mungkin merupakan bagian dari antarmuka jaringan tambahan yang dilampirkan ke Pod.

Untuk panduan implementasi, lihat [Panduan Pengaturan Multus](#) di GitHub.

Plugin CNI alternatif untuk kluster Amazon EKS

Plugin [Amazon VPC CNI untuk Kubernetes adalah satu-satunya plugin](#) CNI yang didukung oleh Amazon EKS dengan node Amazon. EC2 Amazon EKS mendukung kemampuan inti Cilium dan Calico untuk Amazon EKS Hybrid Nodes. Amazon EKS menjalankan Kubernetes upstream, sehingga Anda dapat menginstal plugin CNI alternatif yang kompatibel ke node Amazon EC2 di cluster Anda. Jika Anda memiliki node Fargate di cluster Anda, plugin Amazon VPC CNI untuk Kubernetes sudah

ada di node Fargate Anda. Ini satu-satunya plugin CNI yang dapat Anda gunakan dengan node Fargate. Upaya untuk menginstal plugin CNI alternatif pada node Fargate gagal.

Jika Anda berencana untuk menggunakan plugin CNI alternatif di EC2 node Amazon, kami sarankan Anda mendapatkan dukungan komersial untuk plugin atau memiliki keahlian internal untuk memecahkan masalah dan berkontribusi perbaikan pada proyek plugin CNI.

Amazon EKS memelihara hubungan dengan jaringan mitra yang menawarkan dukungan untuk plugin CNI alternatif yang kompatibel. Untuk detail tentang versi, kualifikasi, dan pengujian yang dilakukan, lihat dokumentasi mitra berikut.

Mitra	Produk	Dokumentasi
Tigera	Calico	Instruksi instalasi
Isovalen	silia	Instruksi instalasi
Juniper	Jaringan Contrail Cloud-Native () CN2	Instruksi instalasi
VMware	Antrea	Instruksi instalasi

Amazon EKS bertujuan untuk memberikan berbagai pilihan untuk mencakup semua kasus penggunaan.

Plugin kebijakan jaringan alternatif yang kompatibel

[Calico](#) adalah solusi yang diadopsi secara luas untuk jaringan kontainer dan keamanan.

Menggunakan Calico di EKS memberikan penegakan kebijakan jaringan yang sepenuhnya sesuai untuk kluster EKS Anda. Selain itu, Anda dapat memilih untuk menggunakan jaringan Calico, yang menghemat alamat IP dari VPC yang mendasarinya. [Calico Cloud](#) meningkatkan fitur Calico Open Source, menyediakan kemampuan keamanan dan observabilitas tingkat lanjut.

Arus lalu lintas ke dan dari Pod dengan grup keamanan terkait tidak tunduk pada penegakan kebijakan jaringan Calico dan terbatas pada penegakan kelompok keamanan Amazon VPC saja.

Jika Anda menggunakan penegakan kebijakan jaringan Calico, kami sarankan Anda mengatur variabel lingkungan `ANNOTATE_POD_IP true` untuk menghindari masalah yang diketahui dengan Kubernetes. Untuk menggunakan fitur ini, Anda harus menambahkan patch izin untuk pod

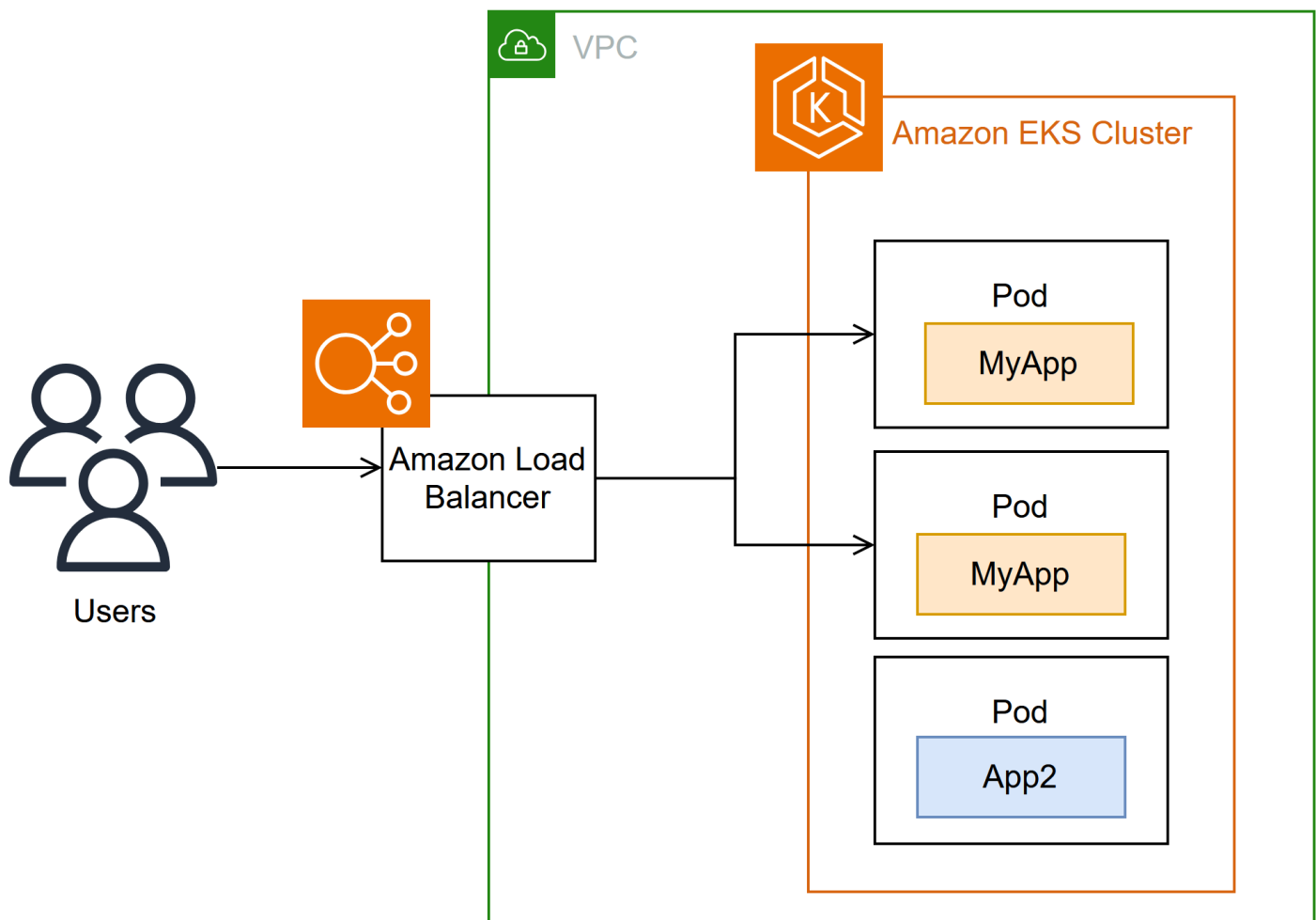
ke file `aws-node ClusterRole`. Perhatikan bahwa menambahkan izin tambahan ke `aws-node DaemonSet` meningkatkan cakupan keamanan untuk plugin. Untuk informasi selengkapnya, lihat [ANNOTATE_POD_IP](#) di repo VPC CNI aktif. GitHub

Pertimbangan untuk Mode Otomatis Amazon EKS

Amazon EKS Auto Mode tidak mendukung plugin CNI alternatif atau plugin kebijakan jaringan. Untuk informasi selengkapnya, lihat [Mode Otomatis EKS](#).

Rute lalu lintas internet dengan AWS Load Balancer Controller

AWS Load Balancer Controller mengelola AWS Elastic Load Balancer untuk kluster Kubernetes. Anda dapat menggunakan controller untuk mengekspos aplikasi cluster Anda ke internet. Pengontrol menyediakan penyeimbang AWS beban yang mengarah ke sumber daya Layanan atau Ingress cluster. Dengan kata lain, controller membuat satu alamat IP atau nama DNS yang menunjuk ke beberapa pod di cluster Anda.



Kontroler mengawasi sumber daya Kubernetes Ingress atau Service. Sebagai tanggapan, ini menciptakan sumber daya AWS Elastic Load Balancing yang sesuai. Anda dapat mengonfigurasi perilaku spesifik penyeimbang beban dengan menerapkan anotasi ke sumber daya Kubernetes. Misalnya, Anda dapat melampirkan grup AWS keamanan ke load balancer menggunakan anotasi.

Pengontrol menyediakan sumber daya berikut:

Kubernetes **Ingress**

LBC membuat [AWS Application Load Balancer \(ALB\)](#) ketika Anda membuat Kubernetes.

Ingress [Tinjau anotasi yang dapat Anda terapkan ke sumber daya Ingress.](#)

Layanan Kubernetes dari tipe **LoadBalancer**

LBC membuat [AWS Network Load Balancer \(NLB\)](#) ketika Anda membuat layanan Kubernetes dari tipe. LoadBalancer [Tinjau anotasi yang dapat Anda terapkan ke sumber daya Layanan.](#)

Di masa lalu, penyeimbang beban jaringan Kubernetes digunakan sebagai target misalnya, tetapi LBC digunakan untuk target IP. Dengan versi AWS Load Balancer Controller 2.3.0 atau yang lebih baru, Anda dapat membuat NLBs menggunakan salah satu jenis target. Untuk informasi selengkapnya terkait tipe-tipe target NLB, lihat [jenis Target](#) di Panduan Pengguna untuk Penyeimbang Beban Jaringan.

Kontroler adalah [proyek sumber terbuka](#) yang dikelola. GitHub

Sebelum menerapkan controller, kami sarankan Anda meninjau prasyarat dan pertimbangan dalam aplikasi [Route dan lalu lintas HTTP dengan Application Load Balancers dan. the section called “Menyeimbangkan beban jaringan”](#) Dalam topik tersebut, Anda akan menerapkan aplikasi sampel yang menyertakan penyeimbang AWS beban.

Instal pengontrol

Anda dapat menggunakan salah satu prosedur berikut untuk menginstal AWS Load Balancer Controller:

- Jika Anda baru mengenal Amazon EKS, kami sarankan Anda menggunakan Helm untuk instalasi karena menyederhanakan instalasi Load AWS Balancer Controller. Untuk informasi selengkapnya, lihat [the section called “Instal dengan Helm”](#).

- Untuk konfigurasi lanjutan, seperti klaster dengan akses jaringan terbatas ke registri kontainer publik, gunakan Kubernetes Manifests. Untuk informasi selengkapnya, lihat [the section called “Instal dengan manifes”](#).

Migrasi dari versi pengontrol yang tidak digunakan lagi

- Jika Anda telah menginstal versi AWS Load Balancer Controller yang sudah usang, lihat. [the section called “Migrasi dari usang”](#)
- Versi usang tidak dapat ditingkatkan. Mereka harus dihapus dan versi AWS Load Balancer Controller saat ini diinstal.
- Versi usang meliputi:
 - AWS ALB Ingress Controller untuk Kubernetes (“Ingress Controller”), pendahulu dari Load Balancer Controller. AWS
 - 0.1.x Versi apa pun dari AWS Load Balancer Controller

Penyedia cloud lama

Kubernetes menyertakan penyedia cloud lama untuk. AWS Penyedia cloud lama mampu menyediakan penyeimbang AWS beban, mirip dengan Load Balancer Controller. AWS Penyedia cloud lama membuat Classic Load Balancer. Jika Anda tidak menginstal AWS Load Balancer Controller, Kubernetes akan secara default menggunakan penyedia cloud lama. Anda harus menginstal AWS Load Balancer Controller dan hindari menggunakan penyedia cloud lama.

Important

Di versi 2.5 dan yang lebih baru, AWS Load Balancer Controller menjadi pengontrol default untuk sumber daya layanan Kubernetes dengan type: `LoadBalancer` dan membuat Network Load AWS Balancer (NLB) untuk setiap layanan. Ini dilakukan dengan membuat webhook yang bermutasi untuk layanan, yang menetapkan `spec.loadBalancerClass` bidang `service.k8s.aws/nlb` untuk layanan baru. type: `LoadBalancer` Anda dapat menonaktifkan fitur ini dan kembali menggunakan [Cloud Provider lama](#) sebagai pengontrol default, dengan menyetel nilai bagan helm ke. `enableServiceMutatorWebhook false` Cluster tidak akan menyediakan Classic Load Balancer baru untuk layanan Anda kecuali Anda menonaktifkan fitur ini. Classic Load Balancer yang ada akan terus bekerja.

Instal AWS Load Balancer Controller dengan Helm

Tip

Dengan Amazon EKS Auto Mode, Anda tidak perlu menginstal atau meningkatkan add-on jaringan. Mode Otomatis mencakup jaringan pod dan kemampuan load balancing. Untuk informasi selengkapnya, lihat [Mode Otomatis EKS](#).

Topik ini menjelaskan cara menginstal AWS Load Balancer Controller menggunakan Helm, manajer paket untuk Kubernetes, dan eksctl. Pengontrol diinstal dengan opsi default. Untuk informasi selengkapnya tentang pengontrol, termasuk detail tentang mengonfigurasinya dengan anotasi, lihat Dokumentasi [AWS Pengontrol Load Balancer aktif](#). GitHub

Pada langkah-langkah berikut, ganti *example values* dengan nilai-nilai Anda sendiri.

Prasyarat

Sebelum memulai tutorial ini, Anda harus menyelesaikan langkah-langkah berikut:

- Buat kluster Amazon EKS. Untuk membuatnya, lihat [Memulai](#).
- Instal [Helm](#) di mesin lokal Anda.
- [Pastikan bahwa plugin Amazon VPC CNI Anda untuk Kubernetes, kube-proxy, dan add-on CoreDNS berada pada versi minimum yang tercantum dalam token akun Layanan.](#)
- Pelajari tentang konsep AWS Elastic Load Balancing. Untuk informasi selengkapnya, silakan lihat [Panduan Pengguna Penyeimbang Beban Elastis](#).
- [Pelajari tentang layanan Kubernetes dan sumber daya ingress.](#)

Pertimbangan

Sebelum melanjutkan dengan langkah-langkah konfigurasi di halaman ini, pertimbangkan hal berikut:

- Kebijakan dan role (AmazonEKSLoadBalancerControllerRole) IAM dapat digunakan kembali di beberapa kluster EKS di akun yang sama. AWS
- Jika Anda menginstal controller pada cluster yang sama di mana role (AmazonEKSLoadBalancerControllerRole) awalnya dibuat, pergi ke [Langkah 2: Instal Load Balancer Controller](#) setelah memverifikasi peran ada.

- Jika Anda menggunakan IAM Roles for Service Accounts (IRSA), IRSA harus disiapkan untuk setiap cluster, dan ARN penyedia OpenID Connect (OIDC) dalam kebijakan kepercayaan peran khusus untuk setiap kluster EKS. Selain itu, jika Anda menginstal pengontrol pada kluster baru dengan yang sudah ada `AmazonEKSLoadBalancerControllerRole`, perbarui kebijakan kepercayaan peran untuk menyertakan penyedia OIDC kluster baru dan buat akun layanan baru dengan anotasi peran yang sesuai. Untuk menentukan apakah Anda sudah memiliki penyedia OIDC, atau membuatnya, lihat [the section called “Penyedia IAM OIDC”](#)

Langkah 1: Buat Peran IAM menggunakan `eksctl`

Langkah-langkah berikut mengacu pada versi rilis AWS Load Balancer Controller v2.13.0. Untuk informasi selengkapnya tentang semua rilis, lihat [Halaman Rilis AWS Load Balancer Controller](#) di GitHub

1. Unduh kebijakan IAM untuk AWS Load Balancer Controller yang memungkinkannya melakukan panggilan atas nama AWS APIs Anda.

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.13.0/docs/install/iam_policy.json
```

- Jika Anda adalah AWS partisi non-standar, seperti Pemerintah atau wilayah Tiongkok, [tinjau kebijakan GitHub dan unduh](#) kebijakan yang sesuai untuk wilayah Anda.
2. Buat kebijakan IAM menggunakan kebijakan yang diunduh di langkah sebelumnya.

```
aws iam create-policy \
  --policy-name AWSLoadBalancerControllerIAMPolicy \
  --policy-document file://iam_policy.json
```

Note

Jika Anda melihat kebijakan di AWS Management Console, konsol akan menampilkan peringatan untuk layanan ELB, tetapi tidak untuk layanan ELB v2. Ini terjadi karena beberapa tindakan dalam kebijakan ada untuk ELB v2, tetapi tidak untuk ELB. Anda dapat mengabaikan peringatan untuk ELB.

3. Ganti nilai untuk nama cluster, kode wilayah, dan ID akun.

```
eksctl create iamserviceaccount \
```

```
--cluster=<cluster-name> \
--namespace=kube-system \
--name=aws-load-balancer-controller \
--attach-policy-arn=arn:aws:iam::<AWS_ACCOUNT_ID>:policy/
AWSLoadBalancerControllerIAMPolicy \
--override-existing-serviceaccounts \
--region <aws-region-code> \
--approve
```

Langkah 2: Instal AWS Load Balancer Controller

1. Tambahkan repositori bagan eks-charts Helm. AWS mempertahankan [repositori ini](#) aktif. GitHub

```
helm repo add eks https://aws.github.io/eks-charts
```

2. Perbarui repo lokal Anda untuk memastikan bahwa Anda memiliki bagan terbaru.

```
helm repo update eks
```

3. Instal AWS Load Balancer Controller.

Jika Anda menerapkan pengontrol ke EC2 node Amazon yang [membatasi akses ke layanan metadata EC2 instans Amazon \(IMDS\)](#), atau jika Anda menerapkan ke Fargate atau Amazon EKS Hybrid Nodes, tambahkan flag berikut ke perintah berikut: `helm`

- `--set region=region-code`
- `--set vpcId=vpc-xxxxxxxx`

Ganti *my-cluster* dengan nama klaster Anda. Dalam perintah berikut, `aws-load-balancer-controller` adalah akun layanan Kubernetes yang Anda buat pada langkah sebelumnya.

Untuk informasi selengkapnya tentang mengonfigurasi bagan helm, lihat [values.yaml](#) di. GitHub

```
helm install aws-load-balancer-controller eks/aws-load-balancer-controller \
-n kube-system \
--set clusterName=my-cluster \
--set serviceAccount.create=false \
--set serviceAccount.name=aws-load-balancer-controller \
```

```
--version 1.13.0
```

⚠ Important

Bagan yang diterapkan tidak menerima pembaruan keamanan secara otomatis. Anda perlu memutakhirkan secara manual ke bagan yang lebih baru ketika bagan tersedia. Saat memutakhirkan, ubah *install* ke upgrade perintah sebelumnya.

`helm install` Perintah secara otomatis menginstal definisi sumber daya kustom (CRDs) untuk pengontrol. `helm upgrade` Perintah tidak. Jika Anda menggunakan, `helm upgrade`, Anda harus menginstal file secara manual CRDs. Jalankan perintah berikut untuk menginstal CRDs:

```
wget https://raw.githubusercontent.com/aws/eks-charts/master/stable/aws-load-balancer-controller/crds/crds.yaml
kubectl apply -f crds.yaml
```

Langkah 3: Verifikasi bahwa pengontrol diinstal

1. Verifikasikan bahwa pengendali telah dipasang.

```
kubectl get deployment -n kube-system aws-load-balancer-controller
```

Contoh output adalah sebagai berikut.

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
aws-load-balancer-controller	2/2	2	2	84s

Anda menerima output sebelumnya jika Anda menerapkan menggunakan Helm. Jika Anda menggunakan manifes Kubernetes, Anda hanya memiliki satu replika.

2. Sebelum menggunakan pengontrol untuk menyediakan AWS sumber daya, kluster Anda harus memenuhi persyaratan tertentu. Lihat informasi yang lebih lengkap di [the section called “Penyeimbangan beban aplikasi”](#) dan [the section called “Menyeimbangan beban jaringan”](#).

Instal AWS Load Balancer Controller dengan manifes

Tip

Dengan Amazon EKS Auto Mode, Anda tidak perlu menginstal atau meningkatkan add-on jaringan. Mode Otomatis mencakup jaringan pod dan kemampuan load balancing. Untuk informasi selengkapnya, lihat [Mode Otomatis EKS](#).

Topik ini menjelaskan cara menginstal controller dengan mengunduh dan menerapkan manifes Kubernetes. Anda dapat melihat [dokumentasi](#) lengkap untuk pengontrol aktif GitHub.

Pada langkah-langkah berikut, ganti *example values* dengan nilai-nilai Anda sendiri.

Prasyarat

Sebelum memulai tutorial ini, Anda harus menyelesaikan langkah-langkah berikut:

- Buat kluster Amazon EKS. Untuk membuatnya, lihat [Memulai](#).
- Instal [Helm](#) di mesin lokal Anda.
- [Pastikan bahwa plugin Amazon VPC CNi Anda untuk Kubernetes, kube-proxy, dan add-on CoreDNS berada pada versi minimum yang tercantum dalam token akun Layanan.](#)
- Pelajari tentang konsep AWS Elastic Load Balancing. Untuk informasi selengkapnya, silakan lihat [Panduan Pengguna Penyeimbang Beban Elastis](#).
- [Pelajari tentang layanan Kubernetes dan sumber daya ingress.](#)

Pertimbangan

Sebelum melanjutkan dengan langkah-langkah konfigurasi di halaman ini, pertimbangkan hal berikut:

- Kebijakan dan role (AmazonEKSLoadBalancerControllerRole) IAM dapat digunakan kembali di beberapa kluster EKS di akun yang sama. AWS
- Jika Anda menginstal controller pada cluster yang sama di mana role (AmazonEKSLoadBalancerControllerRole) awalnya dibuat, pergi ke [Langkah 2: Instal cert-manager](#) setelah memverifikasi peran ada.
- Jika Anda menggunakan IAM Roles for Service Accounts (IRSA), IRSA harus disiapkan untuk setiap cluster, dan ARN penyedia OpenID Connect (OIDC) dalam kebijakan kepercayaan peran

husus untuk setiap kluster EKS. Selain itu, jika Anda menginstal pengontrol pada kluster baru dengan yang sudah ada `AmazonEKSLoadBalancerControllerRole`, perbarui kebijakan kepercayaan peran untuk menyertakan penyedia OIDC kluster baru dan buat akun layanan baru dengan anotasi peran yang sesuai. Untuk menentukan apakah Anda sudah memiliki penyedia OIDC, atau membuatnya, lihat [the section called “Penyedia IAM OIDC”](#)

Langkah 1: Konfigurasikan IAM

Langkah-langkah berikut mengacu pada versi rilis AWS Load Balancer Controller v2.11.0. Untuk informasi selengkapnya tentang semua rilis, lihat [Halaman Rilis AWS Load Balancer Controller](#) di GitHub

1. Unduh kebijakan IAM untuk AWS Load Balancer Controller yang memungkinkannya melakukan panggilan atas nama AWS APIs Anda.

Example

AWS

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.11.0/docs/install/iam_policy.json
```

AWS GovCloud (US)

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.11.0/docs/install/iam_policy_us-gov.json
```

```
mv iam_policy_us-gov.json iam_policy.json
```

2. Buat kebijakan IAM menggunakan kebijakan yang diunduh di langkah sebelumnya.

```
aws iam create-policy \  
  --policy-name AWSLoadBalancerControllerIAMPolicy \  
  --policy-document file://iam_policy.json
```

Note

Jika Anda melihat kebijakan di AWS Management Console, konsol akan menampilkan peringatan untuk layanan ELB, tetapi tidak untuk layanan ELB v2. Ini terjadi karena

beberapa tindakan dalam kebijakan ada untuk ELB v2, tetapi tidak untuk ELB. Anda dapat mengabaikan peringatan untuk ELB.

Example

eksctl

- Ganti *my-cluster* dengan nama cluster Anda, *111122223333* dengan ID akun Anda, dan kemudian jalankan perintah.

```
eksctl create iamserviceaccount \  
  --cluster=my-cluster \  
  --namespace=kube-system \  
  --name=aws-load-balancer-controller \  
  --role-name AmazonEKSLoadBalancerControllerRole \  
  --attach-policy-arn=arn:aws:iam::111122223333:policy/  
AWSLoadBalancerControllerIAMPolicy \  
  --approve
```

AWS CLI and kubectl

- Ambil ID penyedia OIDC cluster Anda dan simpan dalam variabel.

```
oidc_id=$(aws eks describe-cluster --name my-cluster --query  
"cluster.identity.oidc.issuer" --output text | cut -d '/' -f 5)
```

- Tentukan apakah penyedia IAM OIDC dengan ID klaster Anda sudah ada di akun Anda. Anda memerlukan OIDC yang dikonfigurasi untuk cluster dan IAM.

```
aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
```

Jika output dikembalikan, maka Anda sudah memiliki penyedia IAM OIDC untuk cluster Anda. Jika tidak ada output yang dikembalikan, maka Anda harus membuat penyedia IAM OIDC untuk cluster Anda. Untuk informasi selengkapnya, lihat [the section called “Penyedia IAM OIDC”](#).

- Salin konten berikut ke perangkat Anda. Ganti *111122223333* dengan ID akun Anda. Ganti *region-code* dengan AWS Wilayah tempat cluster Anda berada. Ganti

EXAMPLED539D4633E53DE1B71EXAMPLE dengan output yang dikembalikan pada langkah sebelumnya.

```
cat >load-balancer-role-trust-policy.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:oidc-provider/
oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "oidc.eks.region-code.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com",
          "oidc.eks.region-code.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-system:aws-
load-balancer-controller"
        }
      }
    }
  ]
}
EOF
```

d. Buat peran IAM.

```
aws iam create-role \
  --role-name AmazonEKSLoadBalancerControllerRole \
  --assume-role-policy-document file://"load-balancer-role-trust-policy.json"
```

e. Lampirkan kebijakan IAM terkelola Amazon EKS yang diperlukan ke peran IAM. Ganti **111122223333** dengan ID akun Anda.

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::111122223333:policy/
AWSLoadBalancerControllerIAMPolicy \
  --role-name AmazonEKSLoadBalancerControllerRole
```


- f. Salin konten berikut ke perangkat Anda. Ganti **111122223333** dengan ID akun Anda. Setelah mengganti teks, jalankan perintah yang dimodifikasi untuk membuat `aws-load-balancer-controller-service-account.yaml` file.

```
cat >aws-load-balancer-controller-service-account.yaml <<EOF
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    app.kubernetes.io/component: controller
    app.kubernetes.io/name: aws-load-balancer-controller
  name: aws-load-balancer-controller
  namespace: kube-system
  annotations:
    eks.amazonaws.com/role-arn: arn:aws:iam::111122223333:role/
    AmazonEKSLoadBalancerControllerRole
EOF
```

- g. Buat akun layanan Kubernetes di kluster Anda. Akun layanan Kubernetes diberi `aws-load-balancer-controller` anotasi dengan peran IAM yang Anda buat bernama ***AmazonEKSLoadBalancerControllerRole***

```
kubectl apply -f aws-load-balancer-controller-service-account.yaml
```

Langkah 2: Instal **cert-manager**

Instal `cert-manager` menggunakan salah satu metode berikut untuk menyuntikkan konfigurasi sertifikat ke dalam webhook. Untuk informasi selengkapnya, lihat [Memulai](#) di Dokumentasi manajer sertifikat.

Kami merekomendasikan menggunakan registri `quay.io` kontainer untuk menginstal `cert-manager`. Jika node Anda tidak memiliki akses ke registri `quay.io` kontainer, Instal `cert-manager` menggunakan Amazon ECR (lihat di bawah).

Example

Quay.io

- a. Jika node Anda memiliki akses ke registri `quay.io` kontainer, instal `cert-manager` untuk menyuntikkan konfigurasi sertifikat ke dalam webhooks.

```
kubectl apply \
  --validate=false \
  -f https://github.com/jetstack/cert-manager/releases/download/v1.13.5/cert-
manager.yaml
```

Amazon ECR

- a. Instal `cert-manager` menggunakan salah satu metode berikut untuk menyuntikkan konfigurasi sertifikat ke dalam webhook. Untuk informasi selengkapnya, lihat [Memulai](#) di Dokumentasi manajer sertifikat.

- b. Unduh manifes.

```
curl -Lo cert-manager.yaml https://github.com/jetstack/cert-manager/releases/
download/v1.13.5/cert-manager.yaml
```

- c. Tarik gambar berikut dan dorong ke repositori yang dapat diakses oleh node Anda. Untuk informasi lebih lanjut tentang cara menarik, menandai, dan mendorong gambar ke repositori Anda sendiri, lihat [the section called “Salin gambar ke repositori”](#)

```
quay.io/jetstack/cert-manager-cainjector:v1.13.5
quay.io/jetstack/cert-manager-controller:v1.13.5
quay.io/jetstack/cert-manager-webhook:v1.13.5
```

- d. Ganti `quay.io` dalam manifes untuk tiga gambar dengan nama registri Anda sendiri. Perintah berikut mengasumsikan bahwa nama repositori pribadi Anda sama dengan repositori sumber. Ganti `111122223333.dkr.ecr.region-code.amazonaws.com` dengan registri pribadi Anda.

```
sed -i.bak -e 's|quay.io|111122223333.dkr.ecr.region-code.amazonaws.com|' ./cert-
manager.yaml
```

- e. Terapkan manifes.

```
kubectl apply \
  --validate=false \
  -f ./cert-manager.yaml
```

Langkah 3: Instal AWS Load Balancer Controller

1. Unduh spesifikasi pengendali. Untuk informasi selengkapnya tentang pengontrol, lihat [dokumentasi](#) di GitHub.

```
curl -Lo v2_11_0_full.yaml https://github.com/kubernetes-sigs/aws-load-balancer-controller/releases/download/v2.11.0/v2_11_0_full.yaml
```

2. Lakukan pengeditan berikut ke file.

- a. Jika Anda mengunduh `v2_11_0_full.yaml` file, jalankan perintah berikut untuk menghapus `ServiceAccount` bagian dalam manifes. Jika Anda tidak menghapus bagian ini, anotasi yang diperlukan yang Anda buat ke akun layanan pada langkah sebelumnya akan ditimpa. Menghapus bagian ini juga mempertahankan akun layanan yang Anda buat pada langkah sebelumnya jika Anda menghapus pengontrol.

```
sed -i.bak -e '690,698d' ./v2_11_0_full.yaml
```

Jika Anda mengunduh versi file yang berbeda, buka file di editor dan hapus baris berikut.

```
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    app.kubernetes.io/component: controller
    app.kubernetes.io/name: aws-load-balancer-controller
  name: aws-load-balancer-controller
  namespace: kube-system
---
```

- b. Ganti `your-cluster-name` di Deployment spec bagian file dengan nama cluster Anda dengan mengganti `my-cluster` dengan nama cluster Anda.

```
sed -i.bak -e 's|your-cluster-name|my-cluster|' ./v2_11_0_full.yaml
```

- c. Jika node Anda tidak memiliki akses ke repositori gambar Amazon EKS Amazon ECR, maka Anda perlu menarik gambar berikut dan mendorongnya ke repositori yang dapat diakses oleh node Anda. Untuk informasi selengkapnya tentang cara menarik, menandai, dan mendorong gambar ke repositori Anda sendiri, lihat [the section called “Salin gambar ke repositori”](#)

```
public.ecr.aws/eks/aws-load-balancer-controller:v2.11.0
```

Tambahkan nama registri Anda ke manifes. Perintah berikut mengasumsikan bahwa nama repositori pribadi Anda sama dengan repositori sumber dan menambahkan nama registri pribadi Anda ke file. Ganti `111122223333.dkr.ecr.region-code.amazonaws.com` dengan registri Anda. Baris ini mengasumsikan bahwa Anda menamai repositori pribadi Anda sama dengan repositori sumber. Jika tidak, ubah `eks/aws-load-balancer-controller` teks setelah nama registri pribadi Anda ke nama repositori Anda.

```
sed -i.bak -e 's|public.ecr.aws/eks/aws-load-balancer-controller|
111122223333.dkr.ecr.region-code.amazonaws.com/eks/aws-load-balancer-
controller|' ./v2_11_0_full.yaml
```

d. (Diperlukan hanya untuk Fargate atau IMDS Terbatas)

Jika Anda menerapkan pengontrol ke EC2 node Amazon yang [membatasi akses ke layanan metadata EC2 instans Amazon \(IMDS\)](#), atau jika Anda menerapkan ke Fargate atau Amazon EKS Hybrid Nodes, [tambahkan bagian](#) bawah. following parameters - args:

```
[...]
spec:
  containers:
    - args:
      - --cluster-name=your-cluster-name
      - --ingress-class=alb
      - --aws-vpc-id=vpc-xxxxxxx
      - --aws-region=region-code

[...]
```

3. Terapkan file.

```
kubectl apply -f v2_11_0_full.yaml
```

4. Unduh IngressClass dan IngressClassParams manifes ke cluster Anda.

```
curl -Lo v2_11_0_ingclass.yaml https://github.com/kubernetes-sigs/aws-load-balancer-
controller/releases/download/v2.11.0/v2_11_0_ingclass.yaml
```

5. Menerapkan manifes ke klaster Anda.

```
kubectl apply -f v2_11_0_ingclass.yaml
```

Langkah 4: Verifikasi bahwa pengontrol diinstal

1. Verifikasikan bahwa pengendali telah dipasang.

```
kubectl get deployment -n kube-system aws-load-balancer-controller
```

Contoh output adalah sebagai berikut.

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
aws-load-balancer-controller	2/2	2	2	84s

Anda menerima output sebelumnya jika Anda menerapkan menggunakan Helm. Jika Anda menggunakan manifes Kubernetes, Anda hanya memiliki satu replika.

2. Sebelum menggunakan pengontrol untuk menyediakan AWS sumber daya, klaster Anda harus memenuhi persyaratan tertentu. Lihat informasi yang lebih lengkap di [the section called “Penyeimbangan beban aplikasi”](#) dan [the section called “Menyeimbangkan beban jaringan”](#).

Migrasikan aplikasi dari ALB Ingress Controller yang tidak digunakan lagi

Topik ini menjelaskan cara bermigrasi dari versi pengontrol yang tidak digunakan lagi. Lebih khusus lagi, ini menjelaskan cara menghapus versi Load Balancer AWS Controller yang tidak digunakan lagi.

- Versi usang tidak dapat ditingkatkan. Anda harus menghapusnya terlebih dahulu, dan kemudian menginstal versi saat ini.
- Versi usang meliputi:
 - AWS ALB Ingress Controller untuk Kubernetes (“Ingress Controller”), pendahulu dari Load Balancer Controller. AWS
 - 0.1.x Versi apa pun dari AWS Load Balancer Controller

Hapus versi pengontrol yang tidak digunakan lagi

Note

Anda mungkin telah menginstal versi usang menggunakan Helm atau secara manual dengan manifes Kubernetes. Selesaikan prosedur menggunakan alat yang awalnya Anda instal.

1. Jika Anda menginstal bagan `incubator/aws-alb-ingress-controller` Helm, hapus instalannya.

```
helm delete aws-alb-ingress-controller -n kube-system
```

2. Jika Anda memiliki versi `0.1.x` `eks-charts/aws-load-balancer-controller` bagan yang diinstal, hapus instalannya. Pemutakhiran dari `0.1.x` ke versi `1.0.0` tidak berfungsi karena ketidakcocokan dengan versi API webhook.

```
helm delete aws-load-balancer-controller -n kube-system
```

3. Periksa untuk melihat apakah pengendali terpasang saat ini.

```
kubectl get deployment -n kube-system alb-ingress-controller
```

Ini adalah output jika controller tidak diinstal.

```
Error from server (NotFound): deployments.apps "alb-ingress-controller" not found
```

Ini adalah output jika pengontrol diinstal.

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
alb-ingress-controller	1/1	1	1	122d

4. Masukkan perintah berikut untuk menghapus pengendali.

```
kubectl delete -f https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-ingress-controller/v1.1.8/docs/examples/alb-ingress-controller.yaml
kubectl delete -f https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-ingress-controller/v1.1.8/docs/examples/rbac-role.yaml
```

Migrasi ke AWS Load Balancer Controller

Untuk melakukan migrasi dari ALB Ingress Controller untuk Kubernetes ke Load AWS Balancer Controller, Anda perlu:

1. Hapus ALB Ingress Controller (lihat di atas).
2. [Instal AWS Load Balancer Controller.](#)
3. Tambahkan kebijakan tambahan ke Peran IAM yang digunakan oleh Pengontrol AWS Load Balancer. Kebijakan ini memungkinkan LBC untuk mengelola sumber daya yang dibuat oleh ALB Ingress Controller untuk Kubernetes.
4. Unduh kebijakan IAM. Kebijakan ini mengizinkan AWS Load Balancer Controller untuk mengelola sumber daya yang dibuat oleh ALB Ingress Controller untuk Kubernetes. Anda juga dapat [view the policy](#).

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.11.0/docs/install/iam_policy_v1_to_v2_additional.json
```

5. Jika cluster Anda berada di AWS Wilayah AWS GovCloud (AS-Timur) atau AWS GovCloud (AS-Barat), maka ganti `arn:aws:` dengan `arn:aws-us-gov:`

```
sed -i.bak -e 's|arn:aws:|arn:aws-us-gov:|' iam_policy_v1_to_v2_additional.json
```

6. Buat kebijakan IAM dan catat ARN yang dikembalikan.

```
aws iam create-policy \
  --policy-name AWSLoadBalancerControllerAdditionalIAMPolicy \
  --policy-document file://iam_policy_v1_to_v2_additional.json
```

7. Lampirkan kebijakan IAM ke peran IAM yang digunakan oleh Load AWS Balancer Controller. Ganti *your-role-name* dengan nama peran, seperti `AmazonEKSLoadBalancerControllerRole`.

Jika Anda membuat peran menggunakan `eksctl`, maka untuk menemukan nama peran yang dibuat, buka [AWS CloudFormation konsol](#) dan pilih tumpukan `eksctl-my-cluster-addon-iam-serviceaccount-kube-system-aws-load-balancer-controller`. Pilih tab Sumber Daya. Nama peran ada di kolom ID Fisik.

```
aws iam attach-role-policy \
  --role-name your-role-name \
```

```
--policy-arn arn:aws:iam::111122223333:policy/
AWSLoadBalancerControllerAdditionalIAMPolicy
```

Kelola CoreDNS untuk DNS di kluster Amazon EKS

Tip

Dengan Amazon EKS Auto Mode, Anda tidak perlu menginstal atau meningkatkan add-on jaringan. Mode Otomatis mencakup jaringan pod dan kemampuan load balancing. Untuk informasi selengkapnya, lihat [Mode Otomatis EKS](#).

CoreDNS adalah server DNS yang fleksibel dan dapat diperluas yang dapat berfungsi sebagai DNS kluster Kubernetes. Saat Anda meluncurkan kluster Amazon EKS dengan setidaknya satu simpul, dua replika citra CoreDNS di-deploy secara default, terlepas dari jumlah simpul yang di-deploy di kluster Anda. Pod-pod CoreDNS menyediakan resolusi nama untuk semua Pod di dalam kluster. Pod CoreDNS dapat di-deploy ke node Fargate jika kluster Anda menyertakan Profil Fargate dengan namespace yang cocok dengan [namespace](#) untuk CoreDNS. deployment Untuk informasi selengkapnya tentang CoreDNS, lihat [Using CoreDNS for Service Discovery](#) dalam dokumentasi Kubernetes.

Versi CoreDNS

Tabel berikut mencantumkan versi terbaru dari jenis add-on Amazon EKS untuk setiap versi Kubernetes.

Versi Kubernetes	Versi CoreDNS
1.33	v1.12.1-eksbuild.2
1.32	v1.11.4-eksbuild.14
1.31	v1.11.4-eksbuild.14
1.30	v1.11.4-eksbuild.14
1.29	v1.11.4-eksbuild.14

Versi Kubernetes	Versi CoreDNS
1.28	v1.10.1-eksbuild.18
1.27	v1.10.1-eksbuild.18
1.26	v1.9.3-eksbuild.22

Important

Jika Anda mengelola sendiri add-on ini, versi dalam tabel mungkin tidak sama dengan versi yang dikelola sendiri yang tersedia. Untuk informasi selengkapnya tentang memperbarui jenis pengaya yang dikelola sendiri, lihat. [the section called “Pembaruan \(dikelola sendiri\)”](#)

Pertimbangan peningkatan CoreDNS yang penting

- Pembaruan CoreDNS memanfaatkan PodDisruptionBudget a untuk membantu menjaga ketersediaan layanan DNS selama proses pembaruan.
- Untuk meningkatkan stabilitas dan ketersediaan Penerapan CoreDNS, v1.9.3-eksbuild.6 versi dan yang lebih baru v1.10.1-eksbuild.3 dan digunakan dengan file. PodDisruptionBudget Jika Anda telah menerapkan yang sudah adaPodDisruptionBudget, pemutakhiran Anda ke versi ini mungkin gagal. Jika pemutakhiran gagal, menyelesaikan salah satu tugas berikut akan menyelesaikan masalah:
 - Saat melakukan pemutakhiran add-on Amazon EKS, pilih untuk mengganti pengaturan yang ada sebagai opsi resolusi konflik Anda. Jika Anda telah membuat pengaturan khusus lainnya ke Deployment, pastikan untuk mencadangkan pengaturan Anda sebelum memutakhirkan sehingga Anda dapat menerapkan kembali pengaturan kustom Anda yang lain setelah upgrade.
 - Hapus yang sudah ada PodDisruptionBudget dan coba upgrade lagi.
- Dalam versi add-on EKS v1.9.3-eksbuild.3 dan yang lebih baru v1.10.1-eksbuild.6 dan yang lebih baru, CoreDNS Deployment menetapkan readinessProbe untuk menggunakan titik akhir. /ready Titik akhir ini diaktifkan dalam file Corefile konfigurasi untuk CoreDNS.

Jika Anda menggunakan kustomCorefile, Anda harus menambahkan ready plugin ke konfigurasi, sehingga /ready titik akhir aktif di CoreDNS agar probe dapat digunakan.

- Dalam versi add-on EKS v1.9.3-eksbuild.7 dan yang lebih baru v1.10.1-eksbuild.4 dan yang lebih baru, Anda dapat mengubah file. PodDisruptionBudget Anda dapat mengedit add-on dan mengubah pengaturan ini di Pengaturan konfigurasi opsional menggunakan bidang dalam contoh berikut. Contoh ini menunjukkan defaultPodDisruptionBudget.

```
{
  "podDisruptionBudget": {
    "enabled": true,
    "maxUnavailable": 1
  }
}
```

Anda dapat mengatur maxUnavailable atau minAvailable, tetapi Anda tidak dapat mengatur keduanya dalam satu PodDisruptionBudget. Untuk informasi selengkapnya tentang PodDisruptionBudgets, lihat [Menentukan a PodDisruptionBudget dalam dokumentasi](#) Kubernetes.

Perhatikan bahwa jika Anda menyetel enabled ke false, PodDisruptionBudget tidak dihapus. Setelah Anda mengatur bidang ini ke false, Anda harus menghapus PodDisruptionBudget objek. Demikian pula, jika Anda mengedit add-on untuk menggunakan versi add-on yang lebih lama (menurunkan versi add-on) setelah memutakhirkan ke versi dengan a PodDisruptionBudget, add-on tidak dihapus. Untuk menghapus PodDisruptionBudget, Anda dapat menjalankan perintah berikut:

```
kubectl delete poddisruptionbudget coredns -n kube-system
```

- Dalam versi add-on EKS v1.10.1-eksbuild.5 dan yang lebih baru, ubah toleransi default dari node-role.kubernetes.io/master:NoSchedule menjadi node-role.kubernetes.io/control-plane:NoSchedule mematuhi KEP 2067. Untuk informasi lebih lanjut tentang KEP 2067, lihat [KEP-2067: Ganti nama label “master” kubeadm dan taint di Kubernetes Enhancement Proposals](#) () pada. KEPs GitHub

Dalam versi add-on EKS v1.8.7-eksbuild.8 dan yang lebih baru v1.9.3-eksbuild.9 dan yang lebih baru, kedua toleransi diatur agar kompatibel dengan setiap versi Kubernetes.

- Dalam versi add-on EKS v1.9.3-eksbuild.11 v1.10.1-eksbuild.7 dan yang lebih baru, CoreDNS Deployment menetapkan nilai default untuk topologySpreadConstraints. Nilai default memastikan bahwa Pod CoreDNS tersebar di Availability Zone jika ada node di beberapa

Availability Zone yang tersedia. Anda dapat menetapkan nilai kustom yang akan digunakan sebagai pengganti nilai default. Nilai default berikut:

```
topologySpreadConstraints:
  - maxSkew: 1
    topologyKey: topology.kubernetes.io/zone
    whenUnsatisfiable: ScheduleAnyway
    labelSelector:
      matchLabels:
        k8s-app: kube-dns
```

v1.11 CoreDNS meningkatkan pertimbangan

- Dalam versi add-on EKS v1.11.1-eksbuild.4 dan yang lebih baru, gambar kontainer didasarkan pada [gambar dasar minimal](#) yang dikelola oleh Amazon EKS Distro, yang berisi paket minimal dan tidak memiliki cangkang. Untuk informasi selengkapnya, lihat [Amazon EKS Distro](#). Penggunaan dan pemecahan masalah gambar CoreDNS tetap sama.

Buat add-on CoreDNS Amazon EKS

Buat add-on CoreDNS Amazon EKS. Anda harus memiliki cluster sebelum membuat add-on. Untuk informasi selengkapnya, lihat [the section called “Membuat klaster”](#).

1. Lihat versi add-on mana yang diinstal pada cluster Anda.

```
kubectl describe deployment coredns --namespace kube-system | grep coredns: | cut -d : -f 3
```

Contoh output adalah sebagai berikut.

```
v1.10.1-eksbuild.13
```

2. Lihat jenis add-on yang diinstal pada cluster Anda. Bergantung pada alat yang digunakan untuk membuat klaster, saat ini Anda mungkin tidak menginstal jenis add-on Amazon EKS di klaster Anda. Ganti *my-cluster* dengan nama klaster Anda.

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query  
addon.addonVersion --output text
```

Jika nomor versi dikembalikan, Anda memiliki jenis add-on Amazon EKS yang diinstal pada cluster Anda dan tidak perlu menyelesaikan langkah-langkah yang tersisa dalam prosedur ini. Jika kesalahan dikembalikan, Anda tidak memiliki jenis add-on Amazon EKS yang diinstal pada cluster Anda. Selesaikan langkah-langkah yang tersisa dari prosedur ini untuk menginstalnya.

3. Simpan konfigurasi add-on yang Anda instal saat ini.

```
kubectl get deployment coredns -n kube-system -o yaml > aws-k8s-coredns-old.yaml
```

4. Buat add-on menggunakan AWS CLI. Jika Anda ingin menggunakan AWS Management Console atau eksctl untuk membuat add-on, lihat [the section called “Buat add-on”](#) dan tentukan nama coredns add-on. Salin perintah yang mengikuti ke perangkat Anda. Buat modifikasi berikut pada perintah, sesuai kebutuhan, dan kemudian jalankan perintah yang dimodifikasi.

- Ganti *my-cluster* dengan nama klaster Anda.
- Ganti *v1.11.3-eksbuild.1* dengan versi terbaru yang tercantum dalam [tabel versi terbaru](#) untuk versi cluster Anda.

```
aws eks create-addon --cluster-name my-cluster --addon-name coredns --addon-version v1.11.3-eksbuild.1
```

Jika Anda telah menerapkan pengaturan khusus ke add-on saat ini yang bertentangan dengan pengaturan default add-on Amazon EKS, pembuatan mungkin gagal. Jika pembuatan gagal, Anda menerima kesalahan yang dapat membantu Anda menyelesaikan masalah. Atau, Anda dapat menambahkan `--resolve-conflicts OVERWRITE` ke perintah sebelumnya. Hal ini memungkinkan add-on untuk menimpa pengaturan kustom yang ada. Setelah Anda membuat add-on, Anda dapat memperbaruinya dengan pengaturan khusus Anda.

5. Konfirmasikan bahwa versi terbaru dari add-on untuk versi Kubernetes klaster Anda telah ditambahkan ke klaster Anda. Ganti *my-cluster* dengan nama klaster Anda.

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query addon.addonVersion --output text
```

Mungkin perlu beberapa detik untuk menyelesaikan pembuatan add-on.

Contoh output adalah sebagai berikut.

```
v1.11.3-eksbuild.1
```

6. Jika Anda membuat pengaturan khusus untuk add-on asli Anda, sebelum Anda membuat add-on Amazon EKS, gunakan konfigurasi yang Anda simpan di langkah sebelumnya untuk memperbarui add-on Amazon EKS dengan pengaturan khusus Anda. Untuk petunjuk untuk memperbarui add-on, lihat [the section called “Pembaruan \(EKS add-on\)”](#).

Perbarui add-on CoreDNS Amazon EKS

Perbarui jenis add-on Amazon EKS. Jika Anda belum menambahkan add-on Amazon EKS ke cluster Anda, [tambahkan](#) atau lihat [the section called “Pembaruan \(dikelola sendiri\)”](#).

Sebelum Anda mulai, tinjau pertimbangan peningkatan. Untuk informasi selengkapnya, lihat [the section called “Pertimbangan peningkatan CoreDNS yang penting”](#).

1. Lihat versi add-on mana yang diinstal pada cluster Anda. Ganti *my-cluster* dengan nama klaster Anda.

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query  
"addon.addonVersion" --output text
```

Contoh output adalah sebagai berikut.

```
v1.10.1-eksbuild.13
```

Jika versi yang dikembalikan sama dengan versi untuk versi Kubernetes cluster Anda di [tabel versi terbaru](#), maka Anda sudah menginstal versi terbaru di cluster Anda dan tidak perlu menyelesaikan sisa prosedur ini. Jika Anda menerima kesalahan, alih-alih nomor versi dalam output Anda, maka Anda tidak memiliki jenis add-on Amazon EKS yang diinstal pada cluster Anda. Anda perlu [membuat add-on](#) sebelum Anda dapat memperbaruinya dengan prosedur ini.

2. Simpan konfigurasi add-on yang Anda instal saat ini.

```
kubectl get deployment coredns -n kube-system -o yaml > aws-k8s-coredns-old.yaml
```

3. Perbarui add-on Anda menggunakan AWS CLI. Jika Anda ingin menggunakan AWS Management Console atau eksctl memperbarui add-on, lihat [the section called “Perbarui add-on”](#). Salin perintah yang mengikuti ke perangkat Anda. Buat modifikasi berikut pada perintah, sesuai kebutuhan, dan kemudian jalankan perintah yang dimodifikasi.

- Ganti *my-cluster* dengan nama klaster Anda.

- Ganti `v1.11.3-eksbuild.1` dengan versi terbaru yang tercantum dalam [tabel versi terbaru](#) untuk versi cluster Anda.
- `--resolve-conflicts` **PRESERVE** Opsi ini mempertahankan nilai konfigurasi yang ada untuk add-on. Jika Anda telah menetapkan nilai kustom untuk pengaturan add-on, dan Anda tidak menggunakan opsi ini, Amazon EKS menimpa nilai Anda dengan nilai defaultnya. Jika Anda menggunakan opsi ini, kami sarankan untuk menguji setiap bidang dan perubahan nilai pada klaster non-produksi sebelum memperbarui add-on pada cluster produksi Anda. Jika Anda mengubah nilai ini `OVERWRITE`, semua pengaturan diubah ke nilai default Amazon EKS. Jika Anda telah menetapkan nilai kustom untuk setelan apa pun, nilai tersebut mungkin akan ditimpa dengan nilai default Amazon EKS. Jika Anda mengubah nilai ini `none`, Amazon EKS tidak mengubah nilai pengaturan apa pun, tetapi pembaruan mungkin gagal. Jika pembaruan gagal, Anda menerima pesan galat untuk membantu menyelesaikan konflik.
- Jika Anda tidak memperbarui pengaturan konfigurasi, hapus `--configuration-values '{"replicaCount":3}'` dari perintah. Jika Anda memperbarui pengaturan konfigurasi, ganti `"replicaCount":3` dengan pengaturan yang ingin Anda atur. Dalam contoh ini, jumlah replika CoreDNS diatur ke 3. Nilai yang Anda tentukan harus valid untuk skema konfigurasi. Jika Anda tidak tahu skema konfigurasi, jalankan `aws eks describe-addon-configuration --addon-name coredns --addon-version v1.11.3-eksbuild.1`, ganti `v1.11.3-eksbuild.1` dengan nomor versi add-on yang ingin Anda lihat konfigurasinya. Skema dikembalikan dalam output. Jika Anda memiliki konfigurasi kustom yang ada, ingin menghapus semuanya, dan mengatur nilai untuk semua pengaturan kembali ke default Amazon EKS, hapus `"replicaCount":3` dari perintah, sehingga Anda kosong. `{}` Untuk informasi selengkapnya tentang setelan CoreDNS, [lihat Menyesuaikan Layanan DNS di dokumentasi Kubernetes](#).

```
aws eks update-addon --cluster-name my-cluster --addon-name coredns --addon-version v1.11.3-eksbuild.1 \
  --resolve-conflicts PRESERVE --configuration-values '{"replicaCount":3}'
```

Mungkin perlu beberapa detik untuk pembaruan selesai.

4. Konfirmasikan bahwa versi add-on telah diperbarui. Ganti `my-cluster` dengan nama klaster Anda.

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns
```

Mungkin perlu beberapa detik untuk pembaruan selesai.

Contoh output adalah sebagai berikut.

```
{
  "addon": {
    "addonName": "coredns",
    "clusterName": "my-cluster",
    "status": "ACTIVE",
    "addonVersion": "v1.11.3-eksbuild.1",
    "health": {
      "issues": []
    },
    "addonArn": "arn:aws:eks:region:111122223333:addon/my-cluster/coredns/
d2c34f06-1111-2222-1eb0-24f64ce37fa4",
    "createdAt": "2023-03-01T16:41:32.442000+00:00",
    "modifiedAt": "2023-03-01T18:16:54.332000+00:00",
    "tags": {},
    "configurationValues": "{\"replicaCount\":3}"
  }
}
```

Perbarui add-on yang dikelola sendiri CoreDNS Amazon EKS

Important

Sebaiknya tambahkan jenis add-on Amazon EKS ke klaster Anda alih-alih menggunakan jenis add-on yang dikelola sendiri. Jika Anda tidak terbiasa dengan perbedaan antara jenis, lihat [the section called “Add-on Amazon EKS”](#). Untuk informasi selengkapnya tentang menambahkan add-on Amazon EKS ke klaster Anda, lihat [the section called “Buat add-on”](#). Jika Anda tidak dapat menggunakan add-on Amazon EKS, kami mendorong Anda untuk mengirimkan masalah tentang mengapa Anda tidak dapat ke repositori [peta jalan GitHub Containers](#).

Sebelum Anda mulai, tinjau pertimbangan peningkatan. Untuk informasi selengkapnya, lihat [the section called “Pertimbangan peningkatan CoreDNS yang penting”](#).

1. Konfirmasikan bahwa Anda memiliki jenis add-on yang dikelola sendiri yang diinstal pada klaster Anda. Ganti *my-cluster* dengan nama klaster Anda.

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query  
addon.addonVersion --output text
```

Jika pesan kesalahan dikembalikan, Anda memiliki jenis add-on yang dikelola sendiri yang diinstal pada kluster Anda. Selesaikan langkah-langkah yang tersisa dalam prosedur ini. Jika nomor versi dikembalikan, Anda memiliki jenis add-on Amazon EKS yang diinstal pada cluster Anda. Untuk memperbarui jenis add-on Amazon EKS, gunakan prosedur di [Perbarui add-on CoreDNS Amazon EKS](#), daripada menggunakan prosedur ini. Jika Anda tidak terbiasa dengan perbedaan antara jenis add-on, lihat [the section called “Add-on Amazon EKS”](#).

2. Lihat versi gambar kontainer mana yang saat ini diinstal di cluster Anda.

```
kubectl describe deployment coredns -n kube-system | grep Image | cut -d ":" -f 3
```

Contoh output adalah sebagai berikut.

```
v1.8.7-eksbuild.2
```

3. Jika versi CoreDNS Anda saat ini atau v1.5.0 lebih baru, tetapi lebih awal dari versi yang tercantum dalam tabel versi [CoreDNS](#), lewati langkah ini. Jika versi Anda saat ini lebih awal dari v1.5.0, maka Anda perlu memodifikasi CoreDNS ConfigMap untuk menggunakan add-on penerusan, bukan add-on proxy.

- a. Buka ConfigMap dengan perintah berikut.

```
kubectl edit configmap coredns -n kube-system
```

- b. Ganti proxy di baris berikut dengan forward. Simpan file dan keluar dari editor.

```
proxy . /etc/resolv.conf
```

4. Jika Anda awalnya menerapkan kluster Anda di Kubernetes 1.17 atau sebelumnya, Anda mungkin perlu menghapus baris yang dihentikan dari manifes CoreDNS Anda.

 Important

Anda harus menyelesaikan langkah ini sebelum memperbarui ke 1.7.0 versi CoreDNS, tetapi disarankan agar Anda menyelesaikan langkah ini bahkan jika Anda memperbarui ke versi sebelumnya.

- a. Periksa untuk melihat apakah manifes CoreDNS Anda memiliki baris.

```
kubectl get configmap coredns -n kube-system -o jsonpath='{$.data.Corefile}' |  
grep upstream
```

Jika tidak ada output yang dikembalikan, manifes Anda tidak memiliki baris dan Anda dapat melompat ke langkah berikutnya untuk memperbarui CoreDNS. Jika output dikembalikan, maka Anda harus menghapus baris.

- b. Edit ConfigMap dengan perintah berikut, hapus baris dalam file yang memiliki kata `upstream` di dalamnya. Jangan ubah apa pun di dalam file. Setelah baris dihapus, simpan perubahannya.

```
kubectl edit configmap coredns -n kube-system -o yaml
```

5. Ambil versi gambar CoreDNS Anda saat ini:

```
kubectl describe deployment coredns -n kube-system | grep Image
```

Contoh output adalah sebagai berikut.

```
602401143452.dkr.ecr.region-code.amazonaws.com/eks/coredns:v1.8.7-eksbuild.2
```

6. Jika Anda memperbarui ke 1.8.3 CoreDNS atau yang lebih baru, maka Anda perlu menambahkan izin ke `endpointslices` Kubernetes. `system:coredns clusterrole`

```
kubectl edit clusterrole system:coredns -n kube-system
```

Tambahkan baris berikut di bawah baris izin yang ada di `rules` bagian file.

```
[...]  
- apiGroups:
```

```
- discovery.k8s.io
resources:
- endpointslices
verbs:
- list
- watch
[...]
```

7. Perbarui add-on CoreDNS dengan **602401143452** mengganti **region-code** dan dengan nilai dari output yang dikembalikan pada langkah sebelumnya. Ganti **v1.11.3-eksbuild.1** dengan versi CoreDNS yang tercantum dalam tabel versi [terbaru untuk versi Kubernetes](#) Anda.

```
kubectl set image deployment.apps/coredns -n kube-system
coredns=602401143452.dkr.ecr.region-code.amazonaws.com/eks/coredns:v1.11.3-eksbuild.1
```

Contoh output adalah sebagai berikut.

```
deployment.apps/coredns image updated
```

8. Periksa lagi versi gambar kontainer untuk mengonfirmasi bahwa itu diperbarui ke versi yang Anda tentukan pada langkah sebelumnya.

```
kubectl describe deployment coredns -n kube-system | grep Image | cut -d ":" -f 3
```

Contoh output adalah sebagai berikut.

```
v1.11.3-eksbuild.1
```

Menskalakan Pod CoreDNS untuk lalu lintas DNS tinggi

Saat Anda meluncurkan kluster Amazon EKS dengan setidaknya satu node, Deployment dua replika image CoreDNS akan diterapkan secara default, terlepas dari jumlah node yang digunakan di cluster Anda. Pod-pod CoreDNS menyediakan resolusi nama untuk semua Pod di dalam kluster. Aplikasi menggunakan resolusi nama untuk terhubung ke pod dan layanan di cluster serta menghubungkan ke layanan di luar cluster. Karena jumlah permintaan untuk resolusi nama (kueri) dari pod meningkat, Pod CoreDNS dapat kewalahan dan melambat, dan menolak permintaan yang tidak dapat ditangani oleh Pod.

Untuk menangani peningkatan beban pada pod CoreDNS, pertimbangkan sistem penskalaan otomatis untuk CoreDNS. Amazon EKS dapat mengelola penskalaan otomatis Penerapan CoreDNS di CoreDNS versi Eks Add-on. Autoscaler CoreDNS ini terus memantau status cluster, termasuk jumlah node dan core CPU. Berdasarkan informasi tersebut, pengontrol akan secara dinamis menyesuaikan jumlah replika penerapan CoreDNS di cluster EKS. Fitur ini berfungsi untuk v1.9 CoreDNS dan yang lebih baru. Untuk informasi selengkapnya tentang versi mana yang kompatibel dengan CoreDNS Autoscaling, lihat bagian berikut.

Kami merekomendasikan penggunaan fitur ini bersama dengan [praktik terbaik Penskalaan Otomatis Kluster EKS lainnya untuk meningkatkan ketersediaan aplikasi secara keseluruhan dan skalabilitas kluster](#).

Prasyarat

Agar Amazon EKS dapat menskalakan penerapan CoreDNS Anda, ada tiga prasyarat:

- Anda harus menggunakan versi Eks Add-on dari CoreDNS.
- Cluster Anda harus menjalankan setidaknya versi cluster minimum dan versi platform.
- Cluster Anda harus menjalankan setidaknya versi minimum Eks Add-on CoreDNS.

Versi cluster minimum

Penskalaan otomatis CoreDNS dilakukan oleh komponen baru di bidang kontrol cluster, yang dikelola oleh Amazon EKS. Karena itu, Anda harus memutakhirkan cluster Anda ke rilis EKS yang mendukung versi platform minimum yang memiliki komponen baru.

Cluster Amazon EKS baru. Untuk menyebarkan satu, lihat [Memulai](#). Cluster harus menjalankan salah satu versi Kubernetes dan versi platform yang tercantum dalam tabel berikut atau versi yang lebih baru. Perhatikan bahwa versi Kubernetes dan platform yang lebih lambat dari yang terdaftar juga didukung. Anda dapat memeriksa versi Kubernetes Anda saat ini dengan mengganti *my-cluster* perintah berikut dengan nama cluster Anda dan kemudian menjalankan perintah yang dimodifikasi:

```
aws eks describe-cluster --name my-cluster --query cluster.version --output text
```

Versi Kubernetes	Versi platform
1.29.3	eks.7

Versi Kubernetes	Versi platform
1.28.8	eks.13
1.27.12	eks.17
1.26.15	eks.18

 Note

Setiap versi platform dari versi Kubernetes yang lebih baru juga didukung, misalnya versi Kubernetes dari dan seterusnya. 1.30 eks.1

Versi pengaya Minimum EKS

Versi Kubernetes	1.29	1.28	1.27	1.26
	v1.11.1-e ksbuild.9	v1.10.1-e ksbuild.11	v1.10.1-e ksbuild.11	v1.9.3-ek sbuild.15

Mengkonfigurasi penskalaan otomatis CoreDNS di AWS Management Console

1. Pastikan klaster Anda berada pada atau di atas versi cluster minimum.

Amazon EKS meningkatkan cluster antara versi platform dari versi Kubernetes yang sama secara otomatis, dan Anda tidak dapat memulai proses ini sendiri. Sebagai gantinya, Anda dapat meng-upgrade cluster Anda ke versi Kubernetes berikutnya, dan cluster akan ditingkatkan ke versi K8s dan versi platform terbaru.

Versi Kubernetes baru terkadang memperkenalkan perubahan signifikan. Oleh karena itu, kami menyarankan Anda menguji perilaku aplikasi Anda dengan menggunakan klaster terpisah dari versi Kubernetes baru sebelum Anda memperbarui cluster produksi Anda.

Untuk meng-upgrade klaster ke versi Kubernetes baru, ikuti prosedur di [Update cluster yang ada ke versi Kubernetes baru](#).

2. Pastikan Anda memiliki Pengaya EKS untuk CoreDNS, bukan Penerapan CoreDNS yang dikelola sendiri.

Bergantung pada alat yang digunakan untuk membuat kluster, saat ini Anda mungkin tidak menginstal jenis add-on Amazon EKS di cluster Anda. Untuk melihat jenis add-on yang diinstal pada cluster Anda, Anda dapat menjalankan perintah berikut. Ganti `my-cluster` dengan nama kluster Anda.

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query  
addon.addonVersion --output text
```

Jika nomor versi dikembalikan, Anda memiliki jenis add-on Amazon EKS yang diinstal pada cluster Anda dan Anda dapat melanjutkan ke langkah berikutnya. Jika kesalahan dikembalikan, Anda tidak memiliki jenis add-on Amazon EKS yang diinstal pada cluster Anda. Selesaikan langkah-langkah prosedur yang tersisa [Buat add-on CoreDNS Amazon EKS](#) untuk mengganti versi yang dikelola sendiri dengan add-on Amazon EKS.

3. Pastikan bahwa Add-on EKS Anda untuk CoreDNS memiliki versi yang sama atau lebih tinggi dari versi Add-on EKS minimum.

Lihat versi add-on mana yang diinstal pada cluster Anda. Anda dapat memeriksa AWS Management Console atau menjalankan perintah berikut:

```
kubectl describe deployment coredns --namespace kube-system | grep coredns: | cut -  
d : -f 3
```

Contoh output adalah sebagai berikut.

```
v1.10.1-eksbuild.13
```

Bandingkan versi ini dengan versi Eks Add-on minimum di bagian sebelumnya. Jika perlu, tingkatkan Eks Add-on ke versi yang lebih tinggi dengan mengikuti prosedur [Perbarui add-on CoreDNS Amazon EKS](#).

4. Tambahkan konfigurasi penskalaan otomatis ke pengaturan konfigurasi Opsional dari Eks Add-on.
 - a. Buka [konsol Amazon EKS](#).
 - b. Di panel navigasi kiri, pilih Clusters, lalu pilih nama cluster yang ingin Anda konfigurasikan add-on.
 - c. Pilih tab Add-ons.

- d. Pilih kotak di kanan atas kotak add-on CoreDNS dan kemudian pilih Edit.
- e. Pada halaman Konfigurasi CoreDNS:
 - i. Pilih Versi yang ingin Anda gunakan. Kami menyarankan Anda menyimpan versi yang sama dengan langkah sebelumnya, dan memperbarui versi dan konfigurasi dalam tindakan terpisah.
 - ii. Perluas pengaturan konfigurasi opsional.
 - iii. Masukkan kunci JSON **"autoscaling"**: dan nilai objek JSON bersarang dengan kunci **"enabled"**: dan nilai **true** dalam nilai Konfigurasi. Teks yang dihasilkan harus berupa objek JSON yang valid. Jika kunci dan nilai ini adalah satu-satunya data dalam kotak teks, kelilingi kunci dan nilai dengan kurawal kurawal. { } Contoh berikut menunjukkan penskalaan otomatis diaktifkan:

```
{
  "autoScaling": {
    "enabled": true
  }
}
```

- iv. (Opsional) Anda dapat memberikan nilai minimum dan maksimum yang penskalaan otomatis dapat menskalakan jumlah pod CoreDNS.

Contoh berikut menunjukkan penskalaan otomatis diaktifkan dan semua kunci opsional memiliki nilai. Kami merekomendasikan bahwa jumlah minimum pod CoreDNS selalu lebih besar dari 2 untuk memberikan ketahanan bagi layanan DNS di kluster.

```
{
  "autoScaling": {
    "enabled": true,
    "minReplicas": 2,
    "maxReplicas": 10
  }
}
```

- f. Untuk menerapkan konfigurasi baru dengan mengganti pod CoreDNS, pilih Simpan perubahan.

Amazon EKS menerapkan perubahan pada Add-on EKS dengan menggunakan peluncuran Deployment Kubernetes untuk CoreDNS. Anda dapat melacak status peluncuran dalam riwayat Pembaruan add-on di dan dengan AWS Management Console . `kubectl rollout status deployment/coredns --namespace kube-system`

`kubectl rollout` memiliki perintah berikut:

```
kubectl rollout

history  -- View rollout history
pause    -- Mark the provided resource as paused
restart  -- Restart a resource
resume   -- Resume a paused resource
status   -- Show the status of the rollout
undo     -- Undo a previous rollout
```

Jika peluncuran terlalu lama, Amazon EKS akan membatalkan peluncuran, dan pesan dengan jenis Pembaruan Addon dan status Gagal akan ditambahkan ke riwayat Pembaruan add-on. Untuk menyelidiki masalah apa pun, mulailah dari riwayat peluncuran, dan jalankan `kubectl logs` di pod CoreDNS untuk melihat log CoreDNS.

5. Jika entri baru dalam riwayat Pembaruan memiliki status Sukses, maka peluncuran telah selesai dan add-on menggunakan konfigurasi baru di semua pod CoreDNS. Saat Anda mengubah jumlah node dan inti CPU node di cluster, Amazon EKS menskalakan jumlah replika penerapan CoreDNS.

Mengkonfigurasi penskalaan otomatis CoreDNS di Antarmuka Baris Perintah AWS

1. Pastikan klaster Anda berada pada atau di atas versi cluster minimum.

Amazon EKS meningkatkan cluster antara versi platform dari versi Kubernetes yang sama secara otomatis, dan Anda tidak dapat memulai proses ini sendiri. Sebagai gantinya, Anda dapat meng-upgrade cluster Anda ke versi Kubernetes berikutnya, dan cluster akan ditingkatkan ke versi K8s dan versi platform terbaru.

Versi Kubernetes baru terkadang memperkenalkan perubahan signifikan. Oleh karena itu, kami menyarankan Anda menguji perilaku aplikasi Anda dengan menggunakan klaster terpisah dari versi Kubernetes baru sebelum Anda memperbarui cluster produksi Anda.

Untuk meng-upgrade klaster ke versi Kubernetes baru, ikuti prosedur di [Update cluster yang ada ke versi Kubernetes baru](#).

2. Pastikan Anda memiliki Pengaya EKS untuk CoreDNS, bukan Penerapan CoreDNS yang dikelola sendiri.

Bergantung pada alat yang digunakan untuk membuat kluster, saat ini Anda mungkin tidak menginstal jenis add-on Amazon EKS di cluster Anda. Untuk melihat jenis add-on yang diinstal pada cluster Anda, Anda dapat menjalankan perintah berikut. Ganti `my-cluster` dengan nama kluster Anda.

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query  
addon.addonVersion --output text
```

Jika nomor versi dikembalikan, Anda memiliki jenis add-on Amazon EKS yang diinstal pada cluster Anda. Jika kesalahan dikembalikan, Anda tidak memiliki jenis add-on Amazon EKS yang diinstal pada cluster Anda. Selesaikan langkah-langkah prosedur yang tersisa [Buat add-on CoreDNS Amazon EKS](#) untuk mengganti versi yang dikelola sendiri dengan add-on Amazon EKS.

3. Pastikan bahwa Add-on EKS Anda untuk CoreDNS memiliki versi yang sama atau lebih tinggi dari versi Add-on EKS minimum.

Lihat versi add-on mana yang diinstal pada cluster Anda. Anda dapat memeriksa AWS Management Console atau menjalankan perintah berikut:

```
kubectl describe deployment coredns --namespace kube-system | grep coredns: | cut -  
d : -f 3
```

Contoh output adalah sebagai berikut.

```
v1.10.1-eksbuild.13
```

Bandingkan versi ini dengan versi Eks Add-on minimum di bagian sebelumnya. Jika perlu, tingkatkan Eks Add-on ke versi yang lebih tinggi dengan mengikuti prosedur [Perbarui add-on CoreDNS Amazon EKS](#).

4. Tambahkan konfigurasi penskalaan otomatis ke pengaturan konfigurasi Opsional dari Eks Add-on.

Jalankan perintah AWS CLI berikut. Ganti `my-cluster` dengan nama cluster Anda dan peran IAM ARN dengan peran yang Anda gunakan.

```
aws eks update-addon --cluster-name my-cluster --addon-name coredns \  
--resolve-conflicts PRESERVE --configuration-values '{"autoScaling":  
{"enabled":true}}'
```


Amazon EKS menerapkan perubahan pada Add-on EKS dengan menggunakan peluncuran Deployment Kubernetes untuk CoreDNS. Anda dapat melacak status peluncuran dalam riwayat Pembaruan add-on di dan dengan AWS Management Console . `kubectl rollout status deployment/coredns --namespace kube-system`

`kubectl rollout` memiliki perintah berikut:

```
kubectl rollout

history  -- View rollout history
pause   -- Mark the provided resource as paused
restart  -- Restart a resource
resume   -- Resume a paused resource
status   -- Show the status of the rollout
undo     -- Undo a previous rollout
```

Jika peluncuran terlalu lama, Amazon EKS akan membatalkan peluncuran, dan pesan dengan jenis Pembaruan Addon dan status Gagal akan ditambahkan ke riwayat Pembaruan add-on. Untuk menyelidiki masalah apa pun, mulailah dari riwayat peluncuran, dan jalankan `kubectl logs` di pod CoreDNS untuk melihat log CoreDNS.

5. (Opsional) Anda dapat memberikan nilai minimum dan maksimum yang penskalaan otomatis dapat menskalakan jumlah pod CoreDNS.

Contoh berikut menunjukkan penskalaan otomatis diaktifkan dan semua kunci opsional memiliki nilai. Kami merekomendasikan bahwa jumlah minimum pod CoreDNS selalu lebih besar dari 2 untuk memberikan ketahanan bagi layanan DNS di kluster.

```
aws eks update-addon --cluster-name my-cluster --addon-name coredns \
    --resolve-conflicts PRESERVE --configuration-values '{"autoScaling":
{"enabled":true,"minReplicas":2,"maxReplicas":10}}'
```

6. Periksa status pembaruan ke add-on dengan menjalankan perintah berikut:

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns
```

Jika Anda melihat baris ini: `"status": "ACTIVE"`, maka peluncuran telah selesai dan add-on menggunakan konfigurasi baru di semua pod CoreDNS. Saat Anda mengubah jumlah node dan inti CPU node di cluster, Amazon EKS menskalakan jumlah replika penerapan CoreDNS.

Pantau resolusi DNS Kubernetes dengan metrik CoreDNS

CoreDNS sebagai add-on EKS mengekspos metrik dari 9153 CoreDNS pada port dalam format Prometheus dalam layanan. kube-dns Anda dapat menggunakan Prometheus, agen CloudWatch Amazon, atau sistem lain yang kompatibel untuk mengikis (mengumpulkan) metrik ini.

Untuk contoh konfigurasi scrape yang kompatibel dengan Prometheus dan agen, lihat konfigurasi [CloudWatch agen untuk Prometheus CloudWatch](#) di Panduan Pengguna Amazon. CloudWatch

Kelola **kube-proxy** di kluster Amazon EKS

Tip

Dengan Amazon EKS Auto Mode, Anda tidak perlu menginstal atau meningkatkan add-on jaringan. Mode Otomatis mencakup jaringan pod dan kemampuan load balancing. Untuk informasi selengkapnya, lihat [Mode Otomatis EKS](#).

Sebaiknya tambahkan jenis add-on Amazon EKS ke kluster Anda alih-alih menggunakan jenis add-on yang dikelola sendiri. Jika Anda tidak terbiasa dengan perbedaan antara jenis, lihat [the section called “Add-on Amazon EKS”](#). Untuk informasi selengkapnya tentang menambahkan add-on Amazon EKS ke kluster Anda, lihat [the section called “Buat add-on”](#). Jika Anda tidak dapat menggunakan add-on Amazon EKS, kami mendorong Anda untuk mengirimkan masalah tentang mengapa Anda tidak dapat ke repositori [peta jalan GitHub Containers](#).

kube-proxyAdd-on ini digunakan di setiap EC2 node Amazon di kluster Amazon EKS Anda. Ini memelihara aturan jaringan pada node Anda dan memungkinkan komunikasi jaringan ke Pod Anda. Add-on tidak diterapkan ke node Fargate di cluster Anda. Untuk informasi selengkapnya, lihat [kube-proxy](#) di dalam dokumentasi Kubernetes.

Instal sebagai Amazon EKS Add-on

Versi **kube-proxy**

Tabel berikut mencantumkan versi terbaru dari jenis add-on Amazon EKS untuk setiap versi Kubernetes.

Versi Kubernetes	Versi kube-proxy
1.33	v1.33.0-eksbuild.2
1.32	v1.32.5-eksbuild.2
1.31	v1.31.9-eksbuild.2
1.30	v1.30.13-eksbuild.2
1.29	v1.29.15-eksbuild.7
1.28	v1.28.15-eksbuild.22
1.27	v1.27.16-eksbuild.32
1.26	v1.26.15-eksbuild.35

Note

Versi dokumentasi sebelumnya tidak benar. kube-proxy versi v1.28.5, v1.27.9, dan v1.26.12 tidak tersedia.

Jika Anda mengelola sendiri add-on ini, versi dalam tabel mungkin tidak sama dengan versi yang dikelola sendiri yang tersedia.

kube-proxy gambar kontainer

Gambar kube-proxy kontainer didasarkan pada [gambar dasar minimal](#) yang dikelola oleh Amazon EKS Distro, yang berisi paket minimal dan tidak memiliki cangkang. Untuk informasi lebih lanjut, lihat [Amazon EKS Distro](#).

Tabel berikut mencantumkan versi gambar kube-proxy kontainer terkelola mandiri terbaru yang tersedia untuk setiap versi kluster Amazon EKS.

Versi	kube-proxy
1.33	v1.33.0-minimal-eksbuild.2

Versi	kube-proxy
1.32	v1.32.5-minimal-eksbuild.2
1.31	v1.31.9-minimal-eksbuild.2
1.30	v1.30.13-minimal-eksbuild.2
1.29	v1.29.15-minimal-eksbuild.7
1.28	v1.28.15-minimal-eksbuild.22
1.27	v1.27.16-minimal-eksbuild.32
1.26	v1.26.15-minimal-eksbuild.35

Saat [memperbarui jenis add-on Amazon EKS](#), Anda menentukan versi add-on Amazon EKS yang valid, yang mungkin bukan versi yang tercantum dalam tabel ini. Ini karena versi [add-on Amazon EKS](#) tidak selalu cocok dengan versi gambar kontainer yang ditentukan saat memperbarui jenis add-on yang dikelola sendiri. Saat memperbarui jenis add-on yang dikelola sendiri, Anda menentukan versi gambar kontainer yang valid yang tercantum dalam tabel ini.

Perbarui add-on yang dikelola sendiri Kubernetes **kube-proxy**

Important

Sebaiknya tambahkan jenis add-on Amazon EKS ke kluster Anda alih-alih menggunakan jenis add-on yang dikelola sendiri. Jika Anda tidak terbiasa dengan perbedaan antara jenis, lihat [the section called “Add-on Amazon EKS”](#). Untuk informasi selengkapnya tentang menambahkan add-on Amazon EKS ke kluster Anda, lihat [the section called “Buat add-on”](#). Jika Anda tidak dapat menggunakan add-on Amazon EKS, kami mendorong Anda untuk mengirimkan masalah tentang mengapa Anda tidak dapat ke repositori [peta jalan GitHub Containers](#).

Prasyarat

- Sebuah kluster Amazon EKS yang sudah ada. Untuk menyebarkan satu, lihat [Memulai](#).

Pertimbangan

- Kube-proxy pada cluster Amazon EKS memiliki [kompatibilitas dan kebijakan mirroring](#) yang sama dengan Kubernetes. Pelajari cara [Memverifikasi kompatibilitas versi add-on Amazon EKS dengan kluster](#).
1. Konfirmasikan bahwa Anda memiliki jenis add-on yang dikelola sendiri yang diinstal pada kluster Anda. Ganti *my-cluster* dengan nama kluster Anda.

```
aws eks describe-addon --cluster-name my-cluster --addon-name kube-proxy --query  
addon.addonVersion --output text
```

Jika pesan kesalahan dikembalikan, Anda memiliki jenis add-on yang dikelola sendiri yang diinstal pada kluster Anda. Langkah-langkah yang tersisa dalam topik ini adalah untuk memperbarui jenis add-on yang dikelola sendiri. Jika nomor versi dikembalikan, Anda memiliki jenis add-on Amazon EKS yang diinstal pada cluster Anda. Untuk memperbaruinya, gunakan prosedur di [Memperbarui add-on Amazon EKS](#), daripada menggunakan prosedur dalam topik ini. Jika Anda tidak terbiasa dengan perbedaan antara jenis add-on, lihat [the section called “Add-on Amazon EKS”](#).

2. Lihat versi gambar kontainer mana yang saat ini diinstal di cluster Anda.

```
kubectl describe daemonset kube-proxy -n kube-system | grep Image
```

Contoh output adalah sebagai berikut.

```
Image:      602401143452.dkr.ecr.region-code.amazonaws.com/eks/kube-proxy:v1.29.1-  
eksbuild.2
```

Dalam contoh output, *v1.29.1-eksbuild.2* adalah versi yang diinstal pada cluster.

3. Perbarui kube-proxy add-on dengan mengganti *602401143452* dan *region-code* dengan nilai dari output Anda di langkah sebelumnya. Ganti *v1.30.6-eksbuild.3* dengan kube-proxy versi yang tercantum dalam versi [image kontainer kube-proxy dikelola mandiri terbaru yang tersedia untuk setiap tabel versi kluster Amazon EKS](#).

 Important

Manifes untuk setiap jenis gambar berbeda dan tidak kompatibel antara jenis gambar default atau minimal. Anda harus menggunakan jenis gambar yang sama dengan gambar sebelumnya, sehingga titik masuk dan argumen cocok.

```
kubectl set image daemonset.apps/kube-proxy -n kube-system kube-  
proxy=602401143452.dkr.ecr.region-code.amazonaws.com/eks/kube-proxy:v1.30.6-  
eksbuild.3
```

Contoh output adalah sebagai berikut.

```
daemonset.apps/kube-proxy image updated
```

4. Konfirmasikan bahwa versi baru sekarang diinstal pada cluster Anda.

```
kubectl describe daemonset kube-proxy -n kube-system | grep Image | cut -d ":" -f 3
```

Contoh output adalah sebagai berikut.

```
v1.30.0-eksbuild.3
```

5. Jika Anda menggunakan x86 dan Arm node di cluster yang sama dan cluster Anda telah diterapkan sebelum 17 Agustus 2020. Lalu, edit manifes kube-proxy Anda guna menyertakan selektor simpul untuk beberapa arsitektur perangkat keras dengan perintah berikut. Ini adalah operasi satu kali. Setelah menambahkan pemilih ke manifes, Anda tidak perlu menambahkannya setiap kali memperbarui add-on. Jika klaster Anda di-deploy pada atau setelah tanggal 17 Agustus 2020, maka kube-proxy sudah memiliki kemampuan multi-arsitektur.

```
kubectl edit -n kube-system daemonset/kube-proxy
```

Tambahkan selektor simpul berikut ke file di dalam editor dan kemudian simpan file. Untuk contoh tempat untuk menyertakan teks ini dalam editor, lihat file [manifes CNI](#) pada GitHub. Hal ini memungkinkan Kubernetes untuk menarik image hardware yang benar berdasarkan arsitektur hardware node.

```
- key: "kubernetes.io/arch"
  operator: In
  values:
  - amd64
  - arm64
```

6. Jika klaster Anda awalnya dibuat dengan versi Kubernetes 1.14 atau yang lebih baru, maka Anda dapat melewati langkah ini karena kube-proxy sudah menyertakan ini. Affinity Rule Jika Anda awalnya membuat klaster Amazon EKS dengan versi Kubernetes 1.13 atau sebelumnya dan bermaksud menggunakan node Fargate di klaster Anda, maka edit kube-proxy manifes Anda untuk menyertakan NodeAffinity aturan untuk kube-proxy mencegah penjadwalan Pod pada node Fargate. Ini adalah pengeditan satu kali. Setelah Anda menambahkan Affinity Rule ke manifes Anda, Anda tidak perlu menambahkannya setiap kali Anda memperbarui add-on. Edit Anda kube-proxy DaemonSet.

```
kubectl edit -n kube-system daemonset/kube-proxy
```

Tambahkan yang berikut ini Affinity Rule ke DaemonSet spec bagian file di editor dan kemudian simpan file. Untuk contoh tempat untuk menyertakan teks ini dalam editor, lihat file [manifes CNI](#) pada GitHub.

```
- key: eks.amazonaws.com/compute-type
  operator: NotIn
  values:
  - fargate
```

Pelajari cara menerapkan beban kerja dan add-on ke Amazon EKS

Beban kerja Anda di-deploy dalam kontainer, yang di-deploy di Pod di Kubernetes. Sebuah Pod mencakup satu atau lebih kontainer. Biasanya, satu atau beberapa Pod yang menyediakan layanan yang sama digunakan dalam layanan Kubernetes. Setelah menerapkan beberapa Pod yang menyediakan layanan yang sama, Anda dapat:

- [Melihat informasi tentang beban kerja](#) yang berjalan pada setiap kluster menggunakan AWS Management Console.
- [Menskalakan Pod secara vertikal ke atas atau ke bawah dengan Kubernetes Vertical Pod Autoscaler](#).
- [Skala horizontal jumlah Pod yang dibutuhkan untuk memenuhi permintaan naik atau turun dengan Kubernetes Horizontal Pod Autoscaler](#).
- Buat [penyeimbang beban jaringan](#) eksternal (untuk Pod yang dapat diakses internet) atau internal (untuk Pod pribadi) untuk menyeimbangkan lalu lintas jaringan di seluruh Pod. Keseimbangan beban merutekan lalu lintas di Lapisan 4 dari model OSI.
- Buat [Application Load Balancer](#) untuk menyeimbangkan lalu lintas aplikasi di seluruh Pod. Application Load Balancer merutekan lalu lintas di Lapisan 7 dari model OSI.
- Jika Anda baru mengenal Kubernetes, topik ini membantu Anda [Menerapkan aplikasi sampel](#).
- Anda dapat [membatasi alamat IP yang dapat ditugaskan ke layanan](#) dengan externalIPs.

Menyebarkan contoh aplikasi di Linux

Dalam topik ini, Anda menyebarkan aplikasi sampel ke cluster Anda di node linux.

Prasyarat

- Cluster Kubernetes yang sudah ada dengan setidaknya satu node. Jika Anda tidak memiliki kluster Amazon EKS yang ada, Anda dapat menerapkannya menggunakan salah satu panduan di [Memulai](#).
- Kubectl diinstal pada komputer Anda. Untuk informasi selengkapnya, lihat [the section called “Siapkan kubectl dan eksctl”](#).

- `kubectl` dikonfigurasi untuk berkomunikasi dengan cluster Anda. Untuk informasi selengkapnya, lihat [the section called “Akses klaster dengan kubectl”](#).
- Jika Anda berencana untuk menyebarkan beban kerja sampel Anda ke Fargate, maka Anda harus memiliki [profil Fargate](#) yang ada yang menyertakan namespace yang sama yang dibuat dalam tutorial ini, yaitu, kecuali Anda mengubah nama. `eks-sample-app` Jika Anda membuat cluster dengan salah satu panduan masuk [Memulai](#), maka Anda harus membuat profil baru, atau menambahkan namespace ke profil Anda yang ada, karena profil yang dibuat dalam panduan memulai tidak menentukan namespace yang digunakan dalam tutorial ini. VPC Anda juga harus memiliki setidaknya satu subnet pribadi.

Meskipun banyak variabel yang dapat diubah dalam langkah-langkah berikut, kami sarankan hanya mengubah nilai variabel jika ditentukan. Setelah Anda memiliki pemahaman yang lebih baik tentang Pod, penerapan, dan layanan Kubernetes, Anda dapat bereksperimen dengan mengubah nilai lainnya.

Buat namespace

Namespace memungkinkan Anda mengelompokkan sumber daya di Kubernetes. Untuk informasi selengkapnya, lihat [Ruang nama dalam dokumentasi](#) Kubernetes. Jika Anda berencana untuk menerapkan aplikasi sampel Anda untuk [Menyederhanakan manajemen komputasi dengan AWS Fargate](#), pastikan bahwa nilai untuk namespace Define [Pod mana yang menggunakan AWS Fargate](#) saat diluncurkan adalah `eks-sample-app`

```
kubectl create namespace eks-sample-app
```

Buat penerapan Kubernetes

Buat penerapan Kubernetes. Contoh penerapan ini menarik image kontainer dari repositori publik dan menyebarkan tiga replika (masing-masing Pod) ke klaster Anda. Untuk mempelajari lebih lanjut, lihat [Deployment di dokumentasi](#) Kubernetes.

1. Simpan konten berikut ini ke file bernama `eks-sample-deployment.yaml`. Kontainer dalam aplikasi sampel tidak menggunakan penyimpanan jaringan, tetapi Anda mungkin memiliki aplikasi yang perlu. Untuk informasi selengkapnya, lihat [Penyimpanan data aplikasi](#).
 - `amd64` atau `arm64` values bawah `kubernetes.io/arch` kunci berarti bahwa aplikasi dapat digunakan untuk salah satu arsitektur perangkat keras (jika Anda memiliki keduanya di cluster Anda). Ini dimungkinkan karena gambar ini adalah gambar multi-arsitektur, tetapi

tidak semuanya. Anda dapat menentukan arsitektur perangkat keras tempat gambar didukung dengan melihat [detail gambar](#) di repositori tempat Anda menariknya. Saat menerapkan gambar yang tidak mendukung jenis arsitektur perangkat keras, atau yang Anda tidak ingin gambar disebar, hapus jenis itu dari manifes. Untuk informasi selengkapnya, lihat [Label Terkenal, Anotasi, dan Taints di dokumentasi](#) Kubernetes.

- `kubernetes.io/os: linux` nodeSelector Artinya jika Anda memiliki node Linux dan Windows (misalnya) di cluster Anda, gambar hanya akan digunakan ke node Linux. Untuk informasi selengkapnya, lihat [Label Terkenal, Anotasi, dan Taints di dokumentasi](#) Kubernetes.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: eks-sample-linux-deployment
  namespace: eks-sample-app
  labels:
    app: eks-sample-linux-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: eks-sample-linux-app
  template:
    metadata:
      labels:
        app: eks-sample-linux-app
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
              - matchExpressions:
                  - key: kubernetes.io/arch
                    operator: In
                    values:
                      - amd64
                      - arm64
      containers:
        - name: nginx
          image: public.ecr.aws/nginx/nginx:1.23
          ports:
            - name: http
              containerPort: 80
```

```
imagePullPolicy: IfNotPresent
nodeSelector:
  kubernetes.io/os: linux
```

2. Terapkan manifes penerapan ke cluster Anda.

```
kubectl apply -f eks-sample-deployment.yaml
```

Membuat layanan

Sebuah layanan memungkinkan Anda untuk mengakses semua replika melalui satu alamat IP atau nama. Untuk informasi selengkapnya, lihat [Layanan](#) di dokumentasi Kubernetes. Meskipun tidak diimplementasikan dalam aplikasi sampel, jika Anda memiliki aplikasi yang perlu berinteraksi dengan AWS layanan lain, sebaiknya Anda membuat akun layanan Kubernetes untuk Pod Anda, dan mengaitkannya ke AWS akun IAM. Dengan menentukan akun layanan, Pod Anda hanya memiliki izin minimum yang Anda tentukan untuk berinteraksi dengan layanan lain. Untuk informasi selengkapnya, lihat [the section called “Kredensil dengan IRSA”](#).

1. Simpan konten berikut ini ke file bernama `eks-sample-service.yaml`. Kubernetes memberikan layanan alamat IP sendiri yang hanya dapat diakses dari dalam cluster. Untuk mengakses layanan dari luar klaster Anda, gunakan [AWS Load Balancer](#) Controller untuk memuat aplikasi [keseimbangan](#) atau lalu lintas jaringan ke layanan.

```
apiVersion: v1
kind: Service
metadata:
  name: eks-sample-linux-service
  namespace: eks-sample-app
  labels:
    app: eks-sample-linux-app
spec:
  selector:
    app: eks-sample-linux-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

2. Terapkan manifes layanan ke cluster Anda.

```
kubectl apply -f eks-sample-service.yaml
```

Meninjau sumber daya yang dibuat

1. Lihat semua sumber daya yang ada di namespace `eks-sample-app`.

```
kubectl get all -n eks-sample-app
```

Contoh output adalah sebagai berikut.

NAME	READY	STATUS	RESTARTS	AGE
pod/eks-sample-linux-deployment-65b7669776-m6qxz	1/1	Running	0	27m
pod/eks-sample-linux-deployment-65b7669776-mmxxvd	1/1	Running	0	27m
pod/eks-sample-linux-deployment-65b7669776-qzn22	1/1	Running	0	27m

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)
service/eks-sample-linux-service	ClusterIP	10.100.74.8	<none>	80/TCP

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/eks-sample-linux-deployment	3/3	3	3	27m

NAME	DESIRED	CURRENT	READY
replicaset.apps/eks-sample-linux-deployment-776d8f8fd8	3	3	3

Dalam output, Anda melihat layanan dan penerapan yang ditentukan dalam manifes sampel yang diterapkan pada langkah sebelumnya. Anda juga melihat tiga Pod. Ini karena 3 replicas ditentukan dalam manifes sampel. Untuk informasi selengkapnya tentang Pod, lihat [Pod](#) di dokumentasi Kubernetes. Kubernetes secara otomatis membuat replicaset sumber daya, meskipun tidak ditentukan dalam manifes sampel. Untuk informasi selengkapnya `ReplicaSets`, lihat [ReplicaSet](#) di dokumentasi Kubernetes.

Note

Kubernetes mempertahankan jumlah replika yang ditentukan dalam manifes. [Jika ini adalah penerapan produksi dan Anda ingin Kubernetes menskalakan jumlah replika secara horizontal atau menskalakan sumber daya komputasi untuk Pod secara vertikal, gunakan penerapan pod Scale dengan Horizontal Pod Autoscaler dan sumber daya Adjust pod dengan Vertical Pod Autoscaler untuk melakukannya.](#)

2. Lihat detail layanan yang di-deploy.

```
kubectl -n eks-sample-app describe service eks-sample-linux-service
```

Contoh output adalah sebagai berikut.

```
Name:                eks-sample-linux-service
Namespace:           eks-sample-app
Labels:              app=eks-sample-linux-app
Annotations:         <none>
Selector:            app=eks-sample-linux-app
Type:                ClusterIP
IP Families:         <none>
IP:                  10.100.74.8
IPs:                 10.100.74.8
Port:                <unset> 80/TCP
TargetPort:          80/TCP
Endpoints:           192.168.24.212:80,192.168.50.185:80,192.168.63.93:80
Session Affinity:    None
Events:              <none>
```

Pada output sebelumnya, nilai untuk IP: adalah alamat IP unik yang dapat dicapai dari node atau Pod mana pun di dalam cluster, tetapi tidak dapat dijangkau dari luar cluster. Nilai untuk Endpoints adalah alamat IP yang ditetapkan dari dalam VPC Anda ke Pod yang merupakan bagian dari layanan.

3. Lihat detail salah satu Pod yang tercantum dalam output saat Anda [melihat namespace](#) pada langkah sebelumnya. Ganti `776d8f8fd8-78w66` dengan nilai yang dikembalikan untuk salah satu Pod Anda.

```
kubectl -n eks-sample-app describe pod eks-sample-linux-deployment-65b7669776-m6qxz
```

Output contoh disingkat

```

Name:          eks-sample-linux-deployment-65b7669776-m6qxz
Namespace:     eks-sample-app
Priority:       0
Node:          ip-192-168-45-132.us-west-2.compute.internal/192.168.45.132
[...]
IP:            192.168.63.93
IPs:
  IP:          192.168.63.93
Controlled By: ReplicaSet/eks-sample-linux-deployment-65b7669776
[...]
Conditions:
  Type           Status
  Initialized     True
  Ready           True
  ContainersReady True
  PodScheduled    True
[...]
Events:
  Type    Reason      Age    From
  Message
  ----
  -----
  Normal  Scheduled   3m20s  default-scheduler
  Successfully assigned eks-sample-app/eks-sample-linux-deployment-65b7669776-m6qxz to
  ip-192-168-45-132.us-west-2.compute.internal
  [...]

```

Pada output sebelumnya, nilai untuk IP: adalah IP unik yang ditetapkan ke Pod dari blok CIDR yang ditetapkan ke subnet tempat node berada. Jika Anda memilih untuk menetapkan alamat IP Pod dari blok CIDR yang berbeda, Anda dapat mengubah perilaku default. Untuk informasi selengkapnya, lihat [the section called “Jaringan kustom”](#). Anda juga dapat melihat bahwa penjadwal Kubernetes menjadwalkan Pod Node dengan alamat IP. **192.168.45.132**

Tip

Daripada menggunakan baris perintah, Anda dapat melihat banyak detail tentang Pod, layanan, penerapan, dan sumber daya Kubernetes lainnya di AWS Management Console. Untuk informasi selengkapnya, lihat [the section called “Akses sumber daya kluster”](#).

Jalankan shell pada Pod

1. Jalankan shell pada Pod yang Anda jelaskan pada langkah sebelumnya, ganti **65b7669776-m6qxz** dengan ID salah satu Pod Anda.

```
kubectl exec -it eks-sample-linux-deployment-65b7669776-m6qxz -n eks-sample-app -- /bin/bash
```

2. Dari shell Pod, lihat output dari server web yang diinstal dengan deployment Anda pada langkah sebelumnya. Anda hanya perlu menentukan nama layanan. Ini diselesaikan ke alamat IP layanan oleh CoreDNS, yang digunakan dengan cluster Amazon EKS, secara default.

```
curl eks-sample-linux-service
```

Contoh output adalah sebagai berikut.

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
[...]
```

3. Dari shell Pod, lihat server DNS untuk Pod.

```
cat /etc/resolv.conf
```

Contoh output adalah sebagai berikut.

```
nameserver 10.100.0.10
search eks-sample-app.svc.cluster.local svc.cluster.local cluster.local us-
west-2.compute.internal
options ndots:5
```

Pada output sebelumnya, secara otomatis 10.100.0.10 ditetapkan sebagai nameserver untuk semua Pod yang di-deploy ke cluster.

4. Putuskan sambungan dari Pod dengan mengetik `exit`.
5. Setelah selesai dengan aplikasi sampel, Anda dapat menghapus contoh namespace, layanan, dan deployment dengan perintah berikut.

```
kubectl delete namespace eks-sample-app
```

Langkah Berikutnya

Setelah menerapkan aplikasi sampel, Anda mungkin ingin mencoba beberapa latihan berikut:

- [Rute aplikasi dan lalu lintas HTTP dengan Application Load Balancers](#)
- [Rute lalu lintas TCP dan UDP dengan Network Load Balancers](#)

Menyebarkan contoh aplikasi pada Windows

Dalam topik ini, Anda menyebarkan aplikasi sampel ke cluster Anda di node Windows.

Prasyarat

- Cluster Kubernetes yang sudah ada dengan setidaknya satu node. Jika Anda tidak memiliki kluster Amazon EKS yang ada, Anda dapat menerapkannya menggunakan salah satu panduan di [Memulai](#). Anda harus mengaktifkan [dukungan Windows](#) untuk cluster Anda dan setidaknya satu node Amazon EC2 Windows.
- Kubectl diinstal pada komputer Anda. Untuk informasi selengkapnya, lihat [the section called “Siapkan kubectl dan eksctl”](#).
- Kubectl dikonfigurasi untuk berkomunikasi dengan cluster Anda. Untuk informasi selengkapnya, lihat [the section called “Akses klaster dengan kubectl”](#).
- Jika Anda berencana untuk menyebarkan beban kerja sampel Anda ke Fargate, maka Anda harus memiliki [profil Fargate](#) yang ada yang menyertakan namespace yang sama yang dibuat dalam tutorial ini, yaitu, kecuali Anda mengubah nama. eks-sample-app. Jika Anda membuat cluster dengan salah satu panduan masuk [Memulai](#), maka Anda harus membuat profil baru, atau menambahkan namespace ke profil Anda yang ada, karena profil yang dibuat dalam panduan memulai tidak menentukan namespace yang digunakan dalam tutorial ini. VPC Anda juga harus memiliki setidaknya satu subnet pribadi.

Meskipun banyak variabel yang dapat diubah dalam langkah-langkah berikut, kami sarankan hanya mengubah nilai variabel jika ditentukan. Setelah Anda memiliki pemahaman yang lebih baik tentang

Pod, penerapan, dan layanan Kubernetes, Anda dapat bereksperimen dengan mengubah nilai lainnya.

Buat namespace

Namespace memungkinkan Anda untuk mengelompokkan sumber daya di Kubernetes. Untuk informasi selengkapnya, lihat [Ruang nama dalam dokumentasi](#) Kubernetes. Jika Anda berencana untuk menerapkan aplikasi sampel Anda untuk [Menyederhanakan manajemen komputasi dengan AWS Fargate](#), pastikan bahwa nilai untuk namespace Define [Pod mana yang menggunakan AWS Fargate](#) saat diluncurkan adalah `eks-sample-app`

```
kubectl create namespace eks-sample-app
```

Buat penerapan Kubernetes

Contoh penerapan ini menarik image kontainer dari repositori publik dan menyebarkan tiga replika (masing-masing Pod) ke kluster Anda. Untuk mempelajari lebih lanjut, lihat [Deployment di dokumentasi](#) Kubernetes.

1. Simpan konten berikut ini ke file bernama `eks-sample-deployment.yaml`. Kontainer dalam aplikasi sampel tidak menggunakan penyimpanan jaringan, tetapi Anda mungkin memiliki aplikasi yang perlu. Untuk informasi selengkapnya, lihat [Penyimpanan data aplikasi](#).
 - `kubernetes.io/os: windows` nodeSelector Artinya jika Anda memiliki node Windows dan Linux (misalnya) di cluster Anda, gambar hanya akan digunakan ke node Windows. Untuk informasi selengkapnya, lihat [Label Terkenal, Anotasi, dan Taints di dokumentasi](#) Kubernetes.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: eks-sample-windows-deployment
  namespace: eks-sample-app
  labels:
    app: eks-sample-windows-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: eks-sample-windows-app
  template:
    metadata:
```

```

labels:
  app: eks-sample-windows-app
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
          - matchExpressions:
              - key: kubernetes.io/arch
                operator: In
                values:
                  - amd64
  containers:
    - name: windows-server-iis
      image: mcr.microsoft.com/windows/servercore:ltsc2019
      ports:
        - name: http
          containerPort: 80
      imagePullPolicy: IfNotPresent
      command:
        - powershell.exe
        - -command
        - "Add-WindowsFeature Web-Server; Invoke-WebRequest -UseBasicParsing
          -Uri 'https://dotnetbinaries.blob.core.windows.net/servicemonitor/2.0.1.6/
          ServiceMonitor.exe' -OutFile 'C:\\ServiceMonitor.exe'; echo '<html><body><br/
          ><br/><marquee><H1>Hello EKS!!!<H1><marquee></body><html>' > C:\\inetpub\\wwwroot\\
          \\default.html; C:\\ServiceMonitor.exe 'w3svc'; "
      nodeSelector:
        kubernetes.io/os: windows

```

2. Terapkan manifes penerapan ke klaster Anda.

```
kubectl apply -f eks-sample-deployment.yaml
```

Membuat layanan

Sebuah layanan memungkinkan Anda untuk mengakses semua replika melalui satu alamat IP atau nama. Untuk informasi selengkapnya, lihat [Layanan](#) di dokumentasi Kubernetes. Meskipun tidak diimplementasikan dalam aplikasi sampel, jika Anda memiliki aplikasi yang perlu berinteraksi dengan AWS layanan lain, sebaiknya Anda membuat akun layanan Kubernetes untuk Pod Anda, dan mengaitkannya ke AWS akun IAM. Dengan menentukan akun layanan, Pod Anda hanya memiliki izin

minimum yang Anda tentukan untuk berinteraksi dengan layanan lain. Untuk informasi selengkapnya, lihat [the section called “Kredensial dengan IRSA”](#).

1. Simpan konten berikut ini ke file bernama `eks-sample-service.yaml`. Kubernetes memberikan layanan alamat IP sendiri yang hanya dapat diakses dari dalam cluster. Untuk mengakses layanan dari luar kluster Anda, gunakan [AWS Load Balancer](#) Controller untuk memuat aplikasi [keseimbangan](#) atau lalu lintas jaringan ke layanan.

```
apiVersion: v1
kind: Service
metadata:
  name: eks-sample-windows-service
  namespace: eks-sample-app
  labels:
    app: eks-sample-windows-app
spec:
  selector:
    app: eks-sample-windows-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

2. Terapkan manifes layanan ke cluster Anda.

```
kubectl apply -f eks-sample-service.yaml
```

Meninjau sumber daya yang dibuat

1. Lihat semua sumber daya yang ada di namespace `eks-sample-app`.

```
kubectl get all -n eks-sample-app
```

Contoh output adalah sebagai berikut.

NAME	READY	STATUS	RESTARTS	AGE
pod/eks-sample-windows-deployment-65b7669776-m6qxz	1/1	Running	0	27m
pod/eks-sample-windows-deployment-65b7669776-mmxvd	1/1	Running	0	27m
pod/eks-sample-windows-deployment-65b7669776-qzn22	1/1	Running	0	27m

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)
AGE				
service/eks-sample-windows-service	ClusterIP	10.100.74.8	<none>	80/
TCP 32m				

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/eks-sample-windows-deployment	3/3	3	3	27m

NAME	DESIRED	CURRENT	READY
AGE			
replicaset.apps/eks-sample-windows-deployment-776d8f8fd8	3	3	3
27m			

Dalam output, Anda melihat layanan dan penerapan yang ditentukan dalam manifes sampel yang diterapkan pada langkah sebelumnya. Anda juga melihat tiga Pod. Ini karena 3 replicas ditentukan dalam manifes sampel. Untuk informasi selengkapnya tentang Pod, lihat [Pod](#) di dokumentasi Kubernetes. Kubernetes secara otomatis membuat replicaset sumber daya, meskipun tidak ditentukan dalam manifes sampel. Untuk informasi selengkapnya ReplicaSets, lihat [ReplicaSet](#) di dokumentasi Kubernetes.

Note

Kubernetes mempertahankan jumlah replika yang ditentukan dalam manifes. [Jika ini adalah penerapan produksi dan Anda ingin Kubernetes menskalakan jumlah replika secara horizontal atau menskalakan sumber daya komputasi untuk Pod secara vertikal, gunakan penerapan pod Scale dengan Horizontal Pod Autoscaler dan sumber daya Adjust pod dengan Vertical Pod Autoscaler untuk melakukannya.](#)

2. Lihat detail layanan yang di-deploy.

```
kubectl -n eks-sample-app describe service eks-sample-windows-service
```

Contoh output adalah sebagai berikut.

```
Name:          eks-sample-windows-service
Namespace:    eks-sample-app
Labels:       app=eks-sample-windows-app
Annotations:  <none>
Selector:     app=eks-sample-windows-app
Type:         ClusterIP
```

```

IP Families:      <none>
IP:               10.100.74.8
IPs:              10.100.74.8
Port:             <unset> 80/TCP
TargetPort:       80/TCP
Endpoints:        192.168.24.212:80,192.168.50.185:80,192.168.63.93:80
Session Affinity: None
Events:           <none>

```

Pada output sebelumnya, nilai untuk IP: adalah alamat IP unik yang dapat dicapai dari node atau Pod mana pun di dalam cluster, tetapi tidak dapat dijangkau dari luar cluster. Nilai untuk Endpoints adalah alamat IP yang ditetapkan dari dalam VPC Anda ke Pod yang merupakan bagian dari layanan.

3. Lihat detail salah satu Pod yang tercantum dalam output saat Anda [melihat namespace](#) pada langkah sebelumnya. Ganti `776d8f8fd8-78w66` dengan nilai yang dikembalikan untuk salah satu Pod Anda.

```
kubectl -n eks-sample-app describe pod eks-sample-windows-deployment-65b7669776-m6qxz
```

Output contoh disingkat

```

Name:               eks-sample-windows-deployment-65b7669776-m6qxz
Namespace:          eks-sample-app
Priority:            0
Node:               ip-192-168-45-132.us-west-2.compute.internal/192.168.45.132
[...]
IP:                 192.168.63.93
IPs:
  IP:               192.168.63.93
Controlled By:      ReplicaSet/eks-sample-windows-deployment-65b7669776
[...]
Conditions:
  Type              Status
  Initialized        True
  Ready             True
  ContainersReady    True
  PodScheduled       True
[...]
Events:
  Type    Reason    Age    From
  Message

```

```

-----
Normal Scheduled 3m20s default-scheduler
Successfully assigned eks-sample-app/eks-sample-windows-deployment-65b7669776-m6qxz
to ip-192-168-45-132.us-west-2.compute.internal
[...]
```

Pada output sebelumnya, nilai untuk IP: adalah IP unik yang ditetapkan ke Pod dari blok CIDR yang ditetapkan ke subnet tempat node berada. Jika Anda memilih untuk menetapkan alamat IP Pod dari blok CIDR yang berbeda, Anda dapat mengubah perilaku default. Untuk informasi selengkapnya, lihat [the section called “Jaringan kustom”](#). Anda juga dapat melihat bahwa penjadwal Kubernetes menjadwalkan Pod Node dengan alamat IP. **192.168.45.132**

Tip

Daripada menggunakan baris perintah, Anda dapat melihat banyak detail tentang Pod, layanan, penerapan, dan sumber daya Kubernetes lainnya di file. AWS Management Console Untuk informasi selengkapnya, lihat [the section called “Akses sumber daya klaster”](#).

Jalankan shell pada Pod

1. Jalankan shell pada Pod yang Anda jelaskan pada langkah sebelumnya, ganti **65b7669776-m6qxz** dengan ID salah satu Pod Anda.

```
kubectl exec -it eks-sample-windows-deployment-65b7669776-m6qxz -n eks-sample-app -- powershell.exe
```

2. Dari shell Pod, lihat output dari server web yang diinstal dengan deployment Anda pada langkah sebelumnya. Anda hanya perlu menentukan nama layanan. Ini diselesaikan ke alamat IP layanan oleh CoreDNS, yang digunakan dengan cluster Amazon EKS, secara default.

```
Invoke-WebRequest -uri eks-sample-windows-service/default.html -UseBasicParsing
```

Contoh output adalah sebagai berikut.

```
StatusCode      : 200
StatusDescription : OK
```

```
Content      : < h t m l > < b o d y > < b r / > < b r / > < m a r q u e e > < H  
1 > H e l l o  
E K S ! ! ! < H 1 > < m a r q u e e > < / b o d y > < h t m l >
```

3. Dari shell Pod, lihat server DNS untuk Pod.

```
Get-NetIPConfiguration
```

Output dipersingkat

```
InterfaceAlias      : vEthernet  
[...]  
IPv4Address         : 192.168.63.14  
[...]  
DNSServer           : 10.100.0.10
```

Pada output sebelumnya, secara otomatis `10.100.0.10` ditetapkan sebagai server DNS untuk semua Pod yang di-deploy ke cluster.

4. Putuskan sambungan dari Pod dengan mengetik `exit`.
5. Setelah selesai dengan aplikasi sampel, Anda dapat menghapus contoh namespace, layanan, dan deployment dengan perintah berikut.

```
kubectl delete namespace eks-sample-app
```

Langkah Berikutnya

Setelah Anda menerapkan aplikasi sampel, Anda mungkin ingin mencoba beberapa latihan berikut:

- [Rute aplikasi dan lalu lintas HTTP dengan Application Load Balancers](#)
- [Rute lalu lintas TCP dan UDP dengan Network Load Balancers](#)

Sesuaikan sumber daya pod dengan Vertical Pod Autoscaler

Kubernetes [Vertical Pod Autoscaler](#) secara otomatis menyesuaikan reservasi CPU dan memori untuk Pod Anda untuk membantu “ukuran yang tepat” aplikasi Anda. Penyesuaian ini dapat meningkatkan pemanfaatan sumber daya cluster dan membebaskan CPU dan memori untuk Pod lainnya. Topik ini

membantu Anda untuk men-deploy Vertikal Pod Autoscaler untuk klaster Anda dan memverifikasi bahwa itu bekerja.

- Anda memiliki klaster Amazon EKS yang ada. Jika tidak, lihat [Memulai](#).
- Anda telah menginstal Server Metrik Kubernetes. Untuk informasi selengkapnya, tinjau [the section called “Server metrik”](#).
- Anda menggunakan klien `kubectl` yang [dikonfigurasi untuk berkomunikasi dengan klaster Amazon EKS](#).
- 1.1.1 OpenSSL atau yang lebih baru diinstal pada perangkat Anda.

Men-deploy Vertical Pod Autoscaler

Pada bagian ini, Anda men-deploy Vertical Pod Autoscaler untuk klaster Anda.

1. Buka jendela terminal dan arahkan ke direktori tempat Anda ingin mengunduh kode sumber Vertical Pod Autoscaler.
2. Kloning repositori [GitHub kubernetes/autoscaler](https://github.com/kubernetes/autoscaler).

```
git clone https://github.com/kubernetes/autoscaler.git
```

3. Mengubah ke direktori `vertical-pod-autoscaler`.

```
cd autoscaler/vertical-pod-autoscaler/
```

4. (Opsional) Jika Anda telah men-deploy versi lain Vertical Pod Autoscaler, hapus dengan perintah berikut.

```
./hack/vpa-down.sh
```

5. Jika node Anda tidak memiliki akses internet ke registri `registry.k8s.io` kontainer, maka Anda perlu menarik gambar berikut dan mendorongnya ke repositori pribadi Anda sendiri. Untuk informasi selengkapnya tentang cara menarik gambar dan mendorongnya ke repositori pribadi Anda sendiri, lihat [the section called “Salin gambar ke repositori”](#)

```
registry.k8s.io/autoscaling/vpa-admission-controller:0.10.0  
registry.k8s.io/autoscaling/vpa-recommender:0.10.0  
registry.k8s.io/autoscaling/vpa-updater:0.10.0
```


Jika Anda mendorong gambar ke repositori Amazon ECR pribadi, ganti manifes dengan `registry.k8s.io` registri Anda. Ganti **111122223333** dengan ID akun Anda. Ganti **region-code** dengan AWS Wilayah tempat cluster Anda berada. Perintah berikut mengasumsikan bahwa Anda menamai repositori Anda sama dengan nama repositori dalam manifes. Jika Anda menamai repositori Anda sesuatu yang berbeda, maka Anda harus mengubahnya juga.

```
sed -i.bak -e 's/registry.k8s.io/111122223333.dkr.ecr.region-code.amazonaws.com/' ./
deploy/admission-controller-deployment.yaml
sed -i.bak -e 's/registry.k8s.io/111122223333.dkr.ecr.region-code.amazonaws.com/' ./
deploy/recommender-deployment.yaml
sed -i.bak -e 's/registry.k8s.io/111122223333.dkr.ecr.region-code.amazonaws.com/' ./
deploy/updater-deployment.yaml
```

6. Men-deploy Vertical Pod Autoscaler untuk klaster Anda dengan perintah berikut.

```
./hack/vpa-up.sh
```

7. Verifikasi bahwa Pod Vertical Pod Autoscaler telah berhasil dibuat.

```
kubectl get pods -n kube-system
```

Contoh output adalah sebagai berikut.

NAME	READY	STATUS	RESTARTS	AGE
[...]				
metrics-server-8459fc497-kfj8w	1/1	Running	0	83m
vpa-admission-controller-68c748777d-ppspd	1/1	Running	0	7s
vpa-recommender-6fc8c67d85-gljlpl	1/1	Running	0	8s
vpa-updater-786b96955c-bgp9d	1/1	Running	0	8s

Uji pemasangan Vertical Pod Autoscaler

Dalam bagian ini, Anda men-deploy contoh aplikasi untuk memverifikasi bahwa Vertical Pod Autoscaler bekerja.

1. Deploy `hamster.yaml` contoh Vertikal Pod Autoscaler dengan perintah berikut.

```
kubectl apply -f examples/hamster.yaml
```

2. Dapatkan Pod dari aplikasi hamster contoh.

```
kubectl get pods -l app=hamster
```

Contoh output adalah sebagai berikut.

```
hamster-c7d89d6db-rglf5    1/1    Running    0        48s
hamster-c7d89d6db-znvz5    1/1    Running    0        48s
```

3. Jelaskan salah satu Pod untuk melihat cpu dan memory reservasi. Ganti *c7d89d6db-rglf5* dengan salah satu yang IDs dikembalikan dalam output Anda dari langkah sebelumnya.

```
kubectl describe pod hamster-c7d89d6db-rglf5
```

Contoh output adalah sebagai berikut.

```
[...]
Containers:
  hamster:
    Container ID:  docker://
e76c2413fc720ac395c33b64588c82094fc8e5d590e373d5f818f3978f577e24
    Image:          registry.k8s.io/ubuntu-slim:0.1
    Image ID:       docker-pullable://registry.k8s.io/ubuntu-
slim@sha256:b6f8c3885f5880a4f1a7cf717c07242eb4858fdd5a84b5ffe35b1cf680ea17b1
    Port:          <none>
    Host Port:     <none>
    Command:
      /bin/sh
    Args:
      -c
      while true; do timeout 0.5s yes >/dev/null; sleep 0.5s; done
    State:          Running
      Started:      Fri, 27 Sep 2019 10:35:16 -0700
    Ready:          True
    Restart Count:  0
    Requests:
      cpu:          100m
      memory:       50Mi
[...]
```

Anda dapat melihat bahwa Pod asli menyimpan 100 millicpu CPU dan 50 mebibytes memori. Untuk contoh aplikasi ini, 100 millicpu kurang dari yang dibutuhkan Pod untuk dijalankan, jadi CPU-constrained. Ini juga menyimpan memori jauh lebih sedikit dari yang dibutuhkan. vpa-recommender Penerapan Vertical Pod Autoscaler menganalisis Pod hamster untuk melihat apakah persyaratan CPU dan memori sesuai. Jika penyesuaian diperlukan, Pod akan vpa-updater diluncurkan kembali dengan nilai yang diperbarui.

4. Tunggu vpa-updater hingga meluncurkan Pod hamster baru. Ini harus memakan waktu satu atau dua menit. Anda dapat memantau Pod dengan perintah berikut.

Note

Jika Anda tidak yakin bahwa Pod baru telah diluncurkan, bandingkan nama Pod dengan daftar sebelumnya. Saat Pod baru diluncurkan, Anda akan melihat nama Pod baru.

```
kubectl get --watch Pods -l app=hamster
```

5. Saat Pod hamster baru dimulai, jelaskan dan lihat reservasi CPU dan memori yang diperbarui.

```
kubectl describe pod hamster-c7d89d6db-jxgfv
```

Contoh output adalah sebagai berikut.

```
[...]
Containers:
  hamster:
    Container ID:
      docker://2c3e7b6fb7ce0d8c86444334df654af6fb3fc88aad4c5d710eac3b1e7c58f7db
    Image:      registry.k8s.io/ubuntu-slim:0.1
    Image ID:    docker-pullable://registry.k8s.io/ubuntu-
slim@sha256:b6f8c3885f5880a4f1a7cf717c07242eb4858fdd5a84b5ffe35b1cf680ea17b1
    Port:       <none>
    Host Port:   <none>
    Command:
      /bin/sh
    Args:
      -c
      while true; do timeout 0.5s yes >/dev/null; sleep 0.5s; done
    State:      Running
```

```

    Started:      Fri, 27 Sep 2019 10:37:08 -0700
    Ready:        True
    Restart Count: 0
    Requests:
      cpu:        587m
      memory:     262144k
    [...]

```

Pada output sebelumnya, Anda dapat melihat bahwa cpu reservasi meningkat menjadi 587 millicpu, yang lebih dari lima kali nilai aslinya. memoryMeningkat menjadi 262.144 Kilobyte, yaitu sekitar 250 mebibytes, atau lima kali nilai aslinya. Pod ini kurang sumber daya, dan Vertical Pod Autoscaler mengoreksi estimasi dengan nilai yang jauh lebih tepat.

6. Menjelaskan sumber daya hamster-vpa untuk melihat rekomendasi baru.

```
kubectl describe vpa/hamster-vpa
```

Contoh output adalah sebagai berikut.

```

Name:          hamster-vpa
Namespace:     default
Labels:        <none>
Annotations:   kubectl.kubernetes.io/last-applied-configuration:
                {"apiVersion":"autoscaling.k8s.io/v1beta2", "kind":"VerticalPodAutoscaler", "metadata":{"annotations":{}, "name":"hamster-
vpa", "namespace":"d...
API Version:   autoscaling.k8s.io/v1beta2
Kind:          VerticalPodAutoscaler
Metadata:
  Creation Timestamp:  2019-09-27T18:22:51Z
  Generation:         23
  Resource Version:    14411
  Self Link:           /apis/autoscaling.k8s.io/v1beta2/namespaces/default/
verticalpodautoscalers/hamster-vpa
  UID:                d0d85fb9-e153-11e9-ae53-0205785d75b0
Spec:
  Target Ref:
    API Version:  apps/v1
    Kind:         Deployment
    Name:         hamster
Status:
  Conditions:

```

```
Last Transition Time: 2019-09-27T18:23:28Z
Status:              True
Type:                RecommendationProvided
Recommendation:
  Container Recommendations:
    Container Name:  hamster
    Lower Bound:
      Cpu:           550m
      Memory:        262144k
    Target:
      Cpu:           587m
      Memory:        262144k
    Uncapped Target:
      Cpu:           587m
      Memory:        262144k
    Upper Bound:
      Cpu:           21147m
      Memory:        387863636
  Events:            <none>
```

7. Ketika Anda selesai bereksperimen dengan contoh aplikasi, Anda dapat menghapusnya dengan perintah berikut.

```
kubectl delete -f examples/hamster.yaml
```

Menskalakan penerapan pod dengan Horizontal Pod Autoscaler

Kubernetes [Horizontal Pod Autoscaler](#) secara otomatis menskalakan jumlah Pod dalam sebuah deployment, replication controller, atau replica set berdasarkan pemanfaatan CPU resource tersebut. Hal ini dapat membantu aplikasi Anda menskalakan keluar untuk memenuhi peningkatan permintaan atau penskalaan kedalam ketika sumber daya tidak diperlukan, sehingga membebaskan simpul Anda untuk aplikasi lain. Bila Anda mengatur persentase pemanfaatan CPU target, Penskala Otomatis Pod Horizontal menskalakan aplikasi Anda kedalam atau keluar untuk mencoba memenuhi target tersebut.

Penskala Otomatis Pod Horizontal adalah standar sumber daya API di Kubernetes yang hanya mengharuskan agar sumber metrik (seperti server metrik Kubernetes) diinstal pada kluster Amazon EKS Anda agar bisa bekerja. Anda tidak perlu men-deploy atau menginstal Penskala Otomatis Pod Horizontal di kluster Anda untuk mulai menskalakan aplikasi Anda. Untuk informasi selengkapnya, lihat [Penskala Otomatis Pod Horizontal](#) dalam dokumentasi Kubernetes.

Gunakan topik ini untuk mempersiapkan Penskala Otomatis Pod Horizontal untuk kluster Amazon EKS Anda dan untuk memverifikasi bahwa itu bekerja dengan contoh aplikasi.

Note

Topik ini didasarkan pada [panduan autoscaler Horizontal Pod dalam dokumentasi Kubernetes](#).

- Anda masih memiliki kluster Amazon EKS. Jika tidak, lihat [Memulai](#).
- Anda telah menginstal Server Metrik Kubernetes. Untuk informasi selengkapnya, tinjau [the section called “Server metrik”](#).
- Anda menggunakan kubectl klien yang [dikonfigurasi untuk berkomunikasi dengan kluster Amazon EKS](#).

Jalankan aplikasi uji Penskala Otomatis Pod Horizontal

Dalam bagian ini, Anda men-deploy contoh aplikasi untuk memverifikasi bahwa Penskala Otomatis Pod Horizontal bekerja.

Note

Contoh ini didasarkan pada [panduan autoscaler Horizontal Pod dalam dokumentasi Kubernetes](#).

1. Deploy aplikasi web server Apache sederhana dengan perintah berikut.

```
kubectl apply -f https://k8s.io/examples/application/php-apache.yaml
```

Pod server web Apache ini diberi batas CPU 500 millicpu dan melayani pada port 80.

2. Buat sumber daya Penskala Otomatis Pod Horizontal untuk deployment php-apache.

```
kubectl autoscale deployment php-apache --cpu-percent=50 --min=1 --max=10
```

Perintah ini menciptakan autoscaler yang menargetkan 50 persen pemanfaatan CPU untuk deployment, dengan minimal satu Pod dan maksimal sepuluh Pod. Ketika beban CPU rata-rata

lebih rendah dari 50 persen, autoscaler mencoba mengurangi jumlah Pod dalam penerapan, menjadi minimal satu. Ketika beban lebih besar dari 50 persen, autoscaler mencoba untuk meningkatkan jumlah Pod dalam penerapan, hingga maksimal sepuluh. Untuk informasi selengkapnya, lihat [Bagaimana cara HorizontalPodAutoscaler kerja?](#) dalam dokumentasi Kubernetes.

3. Jelaskan penskala otomatis dengan perintah berikut untuk menampilkan detailnya.

```
kubectl get hpa
```

Contoh output adalah sebagai berikut.

NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
php-apache	Deployment/php-apache	0%/50%	1	10	1	51s

Seperti yang Anda lihat, beban CPU saat ini 0%, karena belum ada beban di server. Jumlah Pod sudah berada pada batas terendahnya (satu), sehingga tidak dapat menskalakan.

4. Buat beban untuk server web dengan menjalankan wadah.

```
kubectl run -i \
  --tty load-generator \
  --rm --image=busybox \
  --restart=Never \
  -- /bin/sh -c "while sleep 0.01; do wget -q -O- http://php-apache; done"
```

5. Untuk melihat deployment menskalakan keluar, jalankan perintah berikut secara berkala di terminal yang terpisah dari terminal yang Anda jalankan dari langkah sebelumnya.

```
kubectl get hpa php-apache
```

Contoh output adalah sebagai berikut.

NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
php-apache	Deployment/php-apache	250%/50%	1	10	5	4m44s

Mungkin perlu waktu lebih dari satu menit untuk jumlah replika meningkat. Selama persentase CPU sebenarnya lebih tinggi dari persentase target, maka jumlah replika meningkat, hingga 10. Dalam hal ini, itu 250%, sehingga jumlah REPLICAS terus meningkat.

 Note

Butuh beberapa menit sebelum Anda melihat jumlah replika mencapai maksimum. Jika hanya 6 replika, misalnya, yang diperlukan agar beban CPU tetap pada atau di bawah 50%, maka beban tidak akan berskala melebihi 6 replika.

6. Hentikan beban. Di jendela terminal Anda menghasilkan beban, hentikan beban dengan menahan **Ctrl+C** tombol. Anda dapat melihat skala replika kembali ke 1 dengan menjalankan perintah berikut lagi di terminal tempat Anda menonton penskalaan.

```
kubectl get hpa
```

Contoh output adalah sebagai berikut.

NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
php-apache	Deployment/php-apache	0%/50%	1	10	1	25m

 Note

Kerangka waktu default untuk penskalaan mundur adalah lima menit, sehingga akan memakan waktu sebelum Anda melihat jumlah replika mencapai 1 lagi, bahkan ketika persentase CPU saat ini adalah 0 persen. Kerangka waktu dapat dimodifikasi. Untuk informasi selengkapnya, lihat [Penskala Otomatis Pod Horizontal](#) dalam dokumentasi Kubernetes.

7. Setelah Anda selesai bereksperimen dengan aplikasi sampel Anda, hapus sumber daya php-apache.

```
kubectl delete deployment.apps/php-apache service/php-apache  
horizontalpodautoscaler.autoscaling/php-apache
```


Rute lalu lintas TCP dan UDP dengan Network Load Balancers

Note

Baru: Amazon EKS Auto Mode mengotomatiskan tugas rutin untuk penyeimbangan beban. Untuk informasi selengkapnya, lihat:

- [the section called “Menyebarkan penyeimbang beban”](#)
- [the section called “Buat layanan”](#)

Lalu lintas jaringan seimbang beban L4 pada model OSI. Untuk memuat lalu lintas aplikasi keseimbangan di L7, Anda menerapkan Kubernetes ingress, yang menyediakan Application Load Balancer. Untuk informasi selengkapnya, lihat [the section called “Penyeimbangan beban aplikasi”](#). Untuk mempelajari lebih lanjut tentang perbedaan antara kedua jenis load balancing, lihat fitur [Elastic Load Balancing](#) di AWS situs web.

[Saat Anda membuat tipe Kubernetes ServiceLoadBalancer, pengontrol penyeimbang beban penyedia AWS cloud membuat AWSClassic Load Balancer secara default, tetapi juga dapat membuat Network Load Balancer.](#) AWS Pengontrol ini hanya menerima perbaikan bug kritis di masa mendatang. Untuk informasi selengkapnya tentang penggunaan penyeimbang beban penyedia AWS cloud, lihat [pengontrol penyeimbang beban penyedia AWS cloud](#) di dokumentasi Kubernetes. Penggunaannya tidak tercakup dalam topik ini.

Kami menyarankan Anda menggunakan Controller [AWS Load Balancer versi 2.7.2 atau yang lebih baru, bukan pengontrol penyeimbang](#) beban penyedia AWS cloud. AWS Load Balancer Controller membuat AWS Network Load Balancer, tetapi tidak membuat AWS Classic Load Balancer. Sisa topik ini adalah tentang menggunakan AWS Load Balancer Controller.

AWS Network Load Balancer dapat memuat lalu lintas jaringan keseimbangan ke Pod yang di-deploy ke EC2 IP Amazon dan [target instans, ke target](#) IP AWS Fargate, atau ke Amazon EKS Hybrid Nodes sebagai target IP. Untuk informasi selengkapnya, lihat [AWS Load Balancer Controller](#) di GitHub

Prasyarat

Sebelum Anda dapat memuat lalu lintas jaringan keseimbangan menggunakan AWS Load Balancer Controller, Anda harus memenuhi persyaratan berikut.

- Memiliki klaster yang ada. Jika Anda tidak memiliki cluster yang ada, lihat [Memulai](#) . Jika Anda perlu memperbarui versi klaster yang ada, lihat [the section called “Perbarui versi Kubernetes”](#) .
- Minta AWS Load Balancer Controller diterapkan di cluster Anda. Untuk informasi selengkapnya, lihat [the section called “ AWS Pengontrol Load Balancer”](#) . Kami merekomendasikan versi 2.7.2 atau yang lebih baru.
- Setidaknya satu subnet. Jika beberapa subnet yang diberi tag ditemukan di Availability Zone, controller memilih subnet pertama yang subnetnya ID didahulukan secara leksikografis. Subnet harus memiliki setidaknya delapan alamat IP yang tersedia.
- Jika Anda menggunakan versi AWS Load Balancer Controller 2.1.1 atau versi sebelumnya, subnet harus diberi tag sebagai berikut. Jika menggunakan versi 2.1.2 atau yang lebih baru, tag ini opsional. Anda mungkin ingin menandai subnet jika Anda memiliki beberapa cluster yang berjalan di VPC yang sama, atau beberapa AWS layanan berbagi subnet di VPC, dan ingin lebih banyak kontrol atas tempat penyeimbang beban disediakan untuk setiap cluster. Jika Anda secara eksplisit menentukan subnet IDs sebagai anotasi pada objek layanan, maka Kubernetes dan AWS Load Balancer Controller menggunakan subnet tersebut secara langsung untuk membuat penyeimbang beban. Penandaan subnet tidak diperlukan jika Anda memilih untuk menggunakan metode ini untuk menyediakan penyeimbang beban dan Anda dapat melewati persyaratan penandaan subnet pribadi dan publik berikut. Ganti *my-cluster* dengan nama klaster Anda.
 - Kunci – `kubernetes.io/cluster/<my-cluster>`
 - Nilai – `shared` atau `owned`
- Subnet publik dan pribadi Anda harus memenuhi persyaratan berikut, kecuali jika Anda secara eksplisit menentukan subnet IDs sebagai anotasi pada objek layanan atau ingress. Jika Anda menyediakan load balancer dengan secara eksplisit menentukan subnet IDs sebagai anotasi pada objek service atau ingress, maka Kubernetes dan Load Balancer Controller menggunakan subnet tersebut secara langsung untuk membuat AWS load balancer dan tag berikut tidak diperlukan.
 - Subnet pribadi — Harus ditandai dalam format berikut. Hal ini agar Kubernetes dan Load Balancer Controller tahu bahwa subnet dapat digunakan untuk AWS load balancer internal. Jika Anda menggunakan eksctl atau AWS CloudFormation template Amazon EKS untuk membuat VPC Anda setelah 26 Maret 2020, maka subnet diberi tag dengan tepat saat dibuat. Untuk informasi selengkapnya tentang template-template VPC AWS CloudFormation Amazon EKS, lihat [the section called “Buat VPC”](#) .
 - Kunci – `kubernetes.io/role/internal-elb`
 - Nilai – `1`

- Subnet publik — Harus ditandai dalam format berikut. Hal ini agar Kubernetes tahu untuk hanya menggunakan subnet tersebut untuk penyeimbang beban eksternal alih-alih memilih subnet publik di setiap Availability Zone (berdasarkan urutan leksikografis subnet). IDs Jika Anda menggunakan eksctl atau AWS CloudFormation template Amazon EKS untuk membuat VPC Anda setelah 26 Maret 2020, maka subnet diberi tag dengan tepat saat dibuat. Untuk informasi selengkapnya tentang template AWS CloudFormation VPC Amazon EKS, lihat. [the section called “Buat VPC”](#)
- Kunci – `kubernetes.io/role/elb`
- Nilai – 1

Jika tag peran subnet tidak ditambahkan secara eksplisit, pengontrol layanan Kubernetes memeriksa tabel rute subnet VPC kluster Anda untuk menentukan apakah subnet tersebut bersifat pribadi atau publik. Kami menyarankan agar Anda tidak mengandalkan perilaku ini, dan sebagai gantinya secara eksplisit menambahkan tag peran pribadi atau publik. AWS Load Balancer Controller tidak memeriksa tabel rute, dan mengharuskan tag pribadi dan publik hadir untuk penemuan otomatis yang sukses.

Pertimbangan

- Konfigurasi penyeimbang beban Anda dikendalikan oleh anotasi yang ditambahkan ke manifes untuk layanan Anda. Anotasi layanan berbeda saat menggunakan AWS Load Balancer Controller dibandingkan saat menggunakan AWS pengontrol penyeimbang beban penyedia cloud. Pastikan untuk meninjau [anotasi](#) untuk AWS Load Balancer Controller sebelum menerapkan layanan.
- Saat menggunakan [plugin Amazon VPC CNI untuk Kubernetes](#), Load AWS Balancer Controller dapat memuat keseimbangan ke IP EC2 Amazon atau target instans dan target IP Fargate. Saat menggunakan [plugin CNI yang kompatibel dengan Alternatif](#), pengontrol hanya dapat memuat keseimbangan ke target instans, kecuali Anda menyeimbangkan beban ke Amazon EKS Hybrid Nodes. Untuk node hybrid, pengontrol dapat memuat target IP keseimbangan. Untuk informasi selengkapnya tentang jenis target Network Load Balancer, lihat [Jenis target](#) di Panduan Pengguna untuk Network Load Balancers
- Jika Anda ingin menambahkan tag ke penyeimbang beban saat atau setelah dibuat, tambahkan anotasi berikut dalam spesifikasi layanan Anda. Untuk informasi selengkapnya, lihat [Tag AWS Sumber Daya](#) dalam dokumentasi AWS Load Balancer Controller.

```
service.beta.kubernetes.io/aws-load-balancer-additional-resource-tags
```

- Anda dapat menetapkan [Alamat IP Elastis](#) ke Network Load Balancer dengan menambahkan anotasi berikut. Ganti *example values* dengan alamat IP Elastis Anda. Allocation IDs Jumlah Allocation IDs harus sesuai dengan jumlah subnet yang digunakan untuk penyeimbang beban. Untuk informasi selengkapnya, lihat dokumentasi [AWS Load Balancer Controller](#).

```
service.beta.kubernetes.io/aws-load-balancer-eip-allocations: eipalloc-  
xxxxxxxxxxxxxxxxxxxxx,eipalloc-yyyyyyyyyyyyyyyyyy
```

- Amazon EKS menambahkan satu aturan masuk ke grup keamanan node untuk lalu lintas klien dan satu aturan untuk setiap subnet penyeimbang beban di VPC untuk pemeriksaan kesehatan untuk setiap Network Load Balancer yang Anda buat. Deployment layanan jenis LoadBalancer bisa gagal jika Amazon EKS mencoba untuk membuat aturan yang melebihi kuota untuk jumlah maksimum aturan yang diperbolehkan untuk grup keamanan. Untuk informasi selengkapnya, lihat [Grup keamanan](#) untuk Amazon VPC Anda di Panduan Pengguna Amazon VPC. Pertimbangkan opsi berikut untuk meminimalkan kemungkinan melebihi jumlah aturan maksimum untuk grup keamanan:
 - Meminta peningkatan aturan Anda per kuota grup keamanan. Untuk informasi selengkapnya, lihat [Permintaan peningkatan kuota](#) dalam Panduan Pengguna Service Quotas.
 - Gunakan target IP, bukan target instance. Dengan target IP, Anda dapat berbagi aturan untuk port target yang sama. Anda dapat menentukan subnet penyeimbang beban secara manual dengan anotasi. Untuk informasi selengkapnya, lihat [Anotasi](#) di GitHub.
 - Gunakan ingress, bukan layanan tipeLoadBalancer, untuk mengirim lalu lintas ke layanan Anda. AWS Application Load Balancer membutuhkan aturan yang lebih sedikit daripada Network Load Balancer. Anda dapat berbagi ALB di beberapa ingress. Untuk informasi selengkapnya, lihat [the section called “Penyeimbangan beban aplikasi”](#). Anda tidak dapat membagikan Network Load Balancer di beberapa layanan.
 - Deploy klaster Anda ke beberapa akun.
- Jika Pod Anda berjalan di Windows di cluster Amazon EKS, satu layanan dengan load balancer dapat mendukung hingga 1024 Pod back-end. Setiap Pod memiliki alamat IP uniknya sendiri.
- Kami merekomendasikan hanya membuat Network Load Balancer baru dengan Load AWS Balancer Controller. Mencoba mengganti Network Load Balancer yang ada yang dibuat dengan pengontrol penyeimbang beban penyedia AWS cloud dapat menghasilkan beberapa Network Load Balancer yang dapat menyebabkan downtime aplikasi.

Buat penyeimbang beban jaringan

Anda dapat membuat penyeimbang beban jaringan dengan IP atau target instans.

Buat penyeimbang beban jaringan - Target IP

- Anda dapat menggunakan target IP dengan Pod yang di-deploy ke EC2 node Amazon, Fargate, atau Amazon EKS Hybrid Nodes. Layanan Kubernetes Anda harus dibuat sebagai jenis LoadBalancer. Untuk informasi selengkapnya, lihat [LoadBalancerMengetik](#) dokumentasi Kubernetes.

Untuk membuat penyeimbang beban yang menggunakan target IP, tambahkan anotasi berikut ke manifes layanan dan terapkan layanan Anda. `external` Nilai untuk `aws-load-balancer-type` itulah yang menyebabkan AWS Load Balancer Controller, bukan pengontrol penyeimbang beban penyedia AWS cloud, untuk membuat Network Load Balancer. Anda dapat melihat [sampel layanan manifes](#) dengan anotasi.

```
service.beta.kubernetes.io/aws-load-balancer-type: "external"
service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: "ip"
```

Note

Jika Anda load balancing ke IPv6 Pod, tambahkan anotasi berikut. Anda hanya dapat memuat keseimbangan IPv6 ke target IP, bukan target instance. Tanpa anotasi ini, load balancing selesai. IPv4

```
service.beta.kubernetes.io/aws-load-balancer-ip-address-type: dualstack
```

Network Load Balancers dibuat dengan `internal` `aws-load-balancer-scheme`, secara default. Anda dapat meluncurkan Network Load Balancers di subnet apa pun di VPC kluster Anda, termasuk subnet yang tidak ditentukan saat membuat kluster.

Kubernetes memeriksa tabel rute untuk subnet Anda untuk mengidentifikasi apakah mereka publik atau pribadi. Sub-jaringan publik memiliki rute langsung ke internet menggunakan gateway internet, tetapi subnet privat tidak.

Jika Anda ingin membuat Network Load Balancer di subnet publik untuk memuat saldo ke node Amazon EC2 (Fargate hanya dapat bersifat pribadi), tentukan internet-facing dengan anotasi berikut:

```
service.beta.kubernetes.io/aws-load-balancer-scheme: "internet-facing"
```

Note

`service.beta.kubernetes.io/aws-load-balancer-type: "nlb-ip"` Anotasi masih didukung untuk kompatibilitas mundur. Namun, sebaiknya gunakan anotasi sebelumnya untuk penyeimbang beban baru, bukan `service.beta.kubernetes.io/aws-load-balancer-type: "nlb-ip"`

Important

Jangan mengedit anotasi setelah membuat layanan Anda. Jika Anda perlu memodifikasinya, hapus objek layanan dan buat lagi dengan nilai yang diinginkan untuk anotasi ini.

Buat penyeimbang beban jaringan - Target Instance

- Pengontrol penyeimbang beban penyedia AWS cloud membuat Network Load Balancer hanya dengan target instans. Versi 2.2.0 dan yang lebih baru dari AWS Load Balancer Controller juga membuat Network Load Balancers dengan target instans. Kami merekomendasikan untuk menggunakannya, daripada pengontrol penyeimbang beban penyedia AWS cloud, untuk membuat Network Load Balancer baru. Anda dapat menggunakan target instance Network Load Balancer dengan Pod yang di-deploy ke EC2 node Amazon, tetapi tidak ke Fargate. Untuk memuat lalu lintas jaringan keseimbangan di seluruh Pod yang di-deploy ke Fargate, Anda harus menggunakan target IP.

Untuk menerapkan Network Load Balancer ke subnet pribadi, spesifikasi layanan Anda harus memiliki anotasi berikut. Anda dapat melihat [sampel layanan manifes](#) dengan anotasi `external`. Nilai untuk `aws-load-balancer-type` itulah yang menyebabkan AWS Load Balancer

Controller, bukan pengontrol penyeimbang beban penyedia AWS cloud, untuk membuat Network Load Balancer.

```
service.beta.kubernetes.io/aws-load-balancer-type: "external"
service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: "instance"
```

Network Load Balancers dibuat dengan `internalaws-load-balancer-scheme`, secara default. Untuk Network Load Balancer internal, kluster Amazon EKS Anda harus dikonfigurasi untuk menggunakan setidaknya satu subnet pribadi di VPC Anda. Kubernetes memeriksa tabel rute untuk subnet Anda untuk mengidentifikasi apakah mereka publik atau pribadi. Sub-jaringan publik memiliki rute langsung ke internet menggunakan gateway internet, tetapi subnet privat tidak.

Jika Anda ingin membuat Network Load Balancer di subnet publik untuk memuat saldo ke EC2 node Amazon, tentukan `internet-facing` dengan anotasi berikut:

```
service.beta.kubernetes.io/aws-load-balancer-scheme: "internet-facing"
```

Important

Jangan mengedit anotasi setelah membuat layanan Anda. Jika Anda perlu memodifikasinya, hapus objek layanan dan buat lagi dengan nilai yang diinginkan untuk anotasi ini.

(Opsional) Deploy aplikasi sampel

- Setidaknya satu subnet publik atau pribadi di VPC cluster Anda.
 - Minta AWS Load Balancer Controller diterapkan di cluster Anda. Untuk informasi selengkapnya, lihat [the section called “AWS Pengontrol Load Balancer”](#). Kami merekomendasikan versi 2.7.2 atau yang lebih baru.
1. Jika Anda menerapkan ke Fargate, pastikan Anda memiliki subnet pribadi yang tersedia di VPC Anda dan buat profil Fargate. Jika Anda tidak menerapkan ke Fargate, lewati langkah ini. Anda dapat membuat profil dengan menjalankan perintah berikut atau [AWS Management Console](#) menggunakan nilai yang sama untuk name dan namespace yang ada di perintah. Ganti *example values* dengan milik Anda sendiri.

```
eksctl create fargateprofile \
```

```
--cluster my-cluster \  
--region region-code \  
--name nlb-sample-app \  
--namespace nlb-sample-app
```

2. Deploy aplikasi sampel.

a. Buat namespace untuk aplikasi.

```
kubectl create namespace nlb-sample-app
```

b. Simpan konten berikut ini ke file bernama `sample-deployment.yaml` pada komputer Anda.

```
apiVersion: apps/v1  
kind: Deployment  
metadata:  
  name: nlb-sample-app  
  namespace: nlb-sample-app  
spec:  
  replicas: 3  
  selector:  
    matchLabels:  
      app: nginx  
  template:  
    metadata:  
      labels:  
        app: nginx  
    spec:  
      containers:  
        - name: nginx  
          image: public.ecr.aws/nginx/nginx:1.23  
          ports:  
            - name: tcp  
              containerPort: 80
```

c. Menerapkan manifes ke klaster.

```
kubectl apply -f sample-deployment.yaml
```

3. Buat layanan dengan Network Load Balancer yang menghadap ke internet yang memuat saldo ke target IP.

- a. Simpan konten berikut ke file bernama `sample-service.yaml` file di komputer Anda. Jika Anda menerapkan ke node Fargate, hapus `service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing` baris.

```
apiVersion: v1
kind: Service
metadata:
  name: nlb-sample-service
  namespace: nlb-sample-app
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-type: external
    service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: ip
    service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing
spec:
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
  type: LoadBalancer
  selector:
    app: nginx
```

- b. Menerapkan manifes ke klaster.

```
kubectl apply -f sample-service.yaml
```

4. Verifikasi bahwa layanan telah digunakan.

```
kubectl get svc nlb-sample-service -n nlb-sample-app
```

Contoh output adalah sebagai berikut.

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	AGE
sample-service	LoadBalancer	10.100.240.137	k8s-nlbsampl-nlbsampl-xxxxxxxxxx-xxxxxxxxxxxxxxxxxx.elb.region-code.amazonaws.com	80:32400/TCP 16h

Note

Nilai untuk `10.100.240.137` dan `xxxxxxxxxx - xxxxxxxxxxxxxxxxxxxx` akan berbeda dari output contoh (mereka akan unik untuk penyeimbang beban Anda) dan `us-west-2` mungkin berbeda untuk Anda, tergantung pada AWS Wilayah mana cluster Anda berada.

5. Buka [Amazon EC2 AWS Management Console](#). Pilih Grup Target (di bawah Load Balancing) di panel navigasi kiri. Di kolom Nama, pilih nama grup target di mana nilai di kolom Load balancer cocok dengan sebagian nama di EXTERNAL-IP kolom output pada langkah sebelumnya. Misalnya, Anda akan memilih grup target bernama `k8s-default-samplese-xxxxxxxxxx` jika output Anda sama dengan output sebelumnya. Jenis Target adalah IP karena itu ditentukan dalam manifes layanan sampel.
6. Pilih Grup target Anda, kemudian pilih tab Target. Di bawah Target terdaftar, Anda akan melihat tiga alamat IP replika tiga yang di-deploy di langkah sebelumnya. Tunggu sampai status semua target baik sebelum melanjutkan. Mungkin perlu beberapa menit sebelum semua target berada healthy. Target mungkin dalam unhealthy keadaan sebelum berubah menjadi healthy negara.
7. Kirim lalu lintas ke layanan yang diganti `xxxxxxxxxx-xxxxxxxxxxxxxxxxxx` dan `us-west-2` dengan nilai yang dikembalikan dalam output untuk [langkah sebelumnya](#) untuk EXTERNAL-IP. Jika Anda menerapkan ke subnet pribadi, maka Anda harus melihat halaman dari perangkat dalam VPC Anda, seperti host bastion. Untuk informasi selengkapnya, lihat [Linux Bastion Host di AWS](#).

```
curl k8s-default-samplese-xxxxxxxxxx-xxxxxxxxxxxxxxxxxx.elb.region-code.amazonaws.com
```

Contoh output adalah sebagai berikut.

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
[...]
```

8. Setelah selesai menggunakan contoh deployment, service, dan namespace, hapus saja.

```
kubectl delete namespace nlb-sample-app
```

Rute aplikasi dan lalu lintas HTTP dengan Application Load Balancers

Note

Baru: Amazon EKS Auto Mode mengotomatiskan tugas rutin untuk penyeimbangan beban. Untuk informasi selengkapnya, lihat:

- [the section called “Menyebarkan penyeimbang beban”](#)
- [the section called “Buat kelas ingress”](#)

Saat Anda membuat Kubernetesingress, Application Load AWS Balancer (ALB) disediakan yang memuat keseimbangan lalu lintas aplikasi. Untuk mempelajari lebih lanjut, lihat [Apa itu Application Load Balancer?](#) dalam Panduan Pengguna Application Load Balancers dan [Ingress](#) dalam dokumentasi Kubernetes. ALBs dapat digunakan dengan Pod yang di-deploy ke node atau ke AWS Fargate. Anda dapat men-deploy ALB ke subnet publik atau privat.

Lalu lintas aplikasi seimbang L7 pada model OSI. Untuk memuat lalu lintas jaringan keseimbangan diL4, Anda menerapkan Kubernetes service dari jenisnya. LoadBalancer Jenis ini menyediakan AWS Network Load Balancer. Untuk informasi selengkapnya, lihat [the section called “Menyeimbangkan beban jaringan”](#). Untuk mempelajari lebih lanjut tentang perbedaan antara kedua jenis load balancing, lihat fitur [Elastic Load Balancing](#) di AWS situs web.

Prasyarat

Sebelum Anda dapat menyeimbangkan beban lalu lintas aplikasi ke suatu aplikasi, Anda harus memenuhi persyaratan berikut.

- Memiliki klaster yang ada. Jika Anda tidak memiliki cluster yang ada, lihat [Memulai](#). Jika Anda perlu memperbarui versi klaster yang ada, lihat [the section called “Perbarui versi Kubernetes”](#).
- Minta AWS Load Balancer Controller diterapkan di cluster Anda. Untuk informasi selengkapnya, lihat [the section called “ AWS Pengontrol Load Balancer”](#). Kami merekomendasikan versi 2.7.2 atau yang lebih baru.

- Setidaknya dua subnet harus berada di Availability Zone yang berbeda. AWS Load Balancer Controller memilih satu subnet dari setiap Availability Zone. Ketika beberapa subnet yang diberi tag ditemukan di Availability Zone, controller memilih subnet yang subnetnya ID didahulukan secara leksikografis. Setiap subnet harus memiliki setidaknya delapan alamat IP yang tersedia.

Jika Anda menggunakan beberapa grup keamanan yang dilampirkan ke node pekerja, tepat satu grup keamanan harus diberi tag sebagai berikut. Ganti *my-cluster* dengan nama klaster Anda.

- Kunci – `kubernetes.io/cluster/<my-cluster>`
- Nilai – `shared` atau `owned`
- Jika Anda menggunakan versi AWS Load Balancer Controller 2.1.1 atau versi sebelumnya, subnet harus diberi tag dalam format berikut. Jika Anda menggunakan versi 2.1.2 atau yang lebih baru, penandaan bersifat opsional. Namun, kami menyarankan Anda menandai subnet jika salah satu dari yang berikut ini terjadi. Anda memiliki beberapa cluster yang berjalan di VPC yang sama, atau memiliki AWS beberapa layanan yang berbagi subnet dalam VPC. Atau, Anda ingin lebih banyak kontrol di mana penyeimbang beban disediakan untuk setiap cluster. Ganti *my-cluster* dengan nama klaster Anda.
 - Kunci – `kubernetes.io/cluster/<my-cluster>`
 - Nilai – `shared` atau `owned`
- Subnet publik dan pribadi Anda harus memenuhi persyaratan berikut. Ini kecuali Anda secara eksplisit menentukan subnet IDs sebagai anotasi pada layanan atau objek ingress. Misalnya Anda menyediakan penyeimbang beban dengan secara eksplisit menentukan subnet IDs sebagai anotasi pada objek layanan atau ingress. Dalam situasi ini, Kubernetes dan pengontrol penyeimbang AWS beban menggunakan subnet tersebut secara langsung untuk membuat penyeimbang beban dan tag berikut tidak diperlukan.
 - Subnet pribadi — Harus ditandai dalam format berikut. Hal ini agar Kubernetes dan pengontrol AWS load balancer mengetahui bahwa subnet dapat digunakan untuk penyeimbang beban internal. Jika Anda menggunakan eksctl atau AWS CloudFormation template Amazon EKS untuk membuat VPC Anda setelah 26 Maret 2020, subnet diberi tag dengan tepat saat dibuat. Untuk informasi selengkapnya tentang template AWS CloudFormation VPC Amazon EKS, lihat [the section called “Buat VPC”](#)
 - Kunci – `kubernetes.io/role/internal-elb`
 - Nilai – `1`
 - Subnet publik — Harus ditandai dalam format berikut. Ini agar Kubernetes tahu hanya menggunakan subnet yang ditentukan untuk penyeimbang beban eksternal. Dengan cara ini, Kubernetes tidak memilih subnet publik di setiap Availability Zone (secara leksikografis

berdasarkan subnet ID mereka). Jika Anda menggunakan eksctl atau AWS CloudFormation template Amazon EKS untuk membuat VPC Anda setelah 26 Maret 2020, subnet diberi tag dengan tepat saat dibuat. Untuk informasi selengkapnya tentang template AWS CloudFormation VPC Amazon EKS, lihat [the section called “Buat VPC”](#)

- Kunci – `kubernetes.io/role/elb`
- Nilai – 1

Jika tag peran subnet tidak ditambahkan secara eksplisit, pengontrol layanan Kubernetes akan memeriksa tabel rute subnet VPC kluster Anda. Ini untuk menentukan apakah subnet bersifat pribadi atau publik. Kami menyarankan Anda untuk tidak bergantung pada perilaku ini. Sebaliknya, tambahkan tag peran pribadi atau publik secara eksplisit. AWS Load Balancer Controller tidak memeriksa tabel rute. Ini juga membutuhkan tag pribadi dan publik untuk hadir untuk penemuan auto yang sukses.

- [AWS Load Balancer Controller](#) membuat ALBs dan AWS sumber daya pendukung yang diperlukan setiap kali resource ingress Kubernetes dibuat di kluster dengan anotasi. `kubernetes.io/ingress.class: alb` Sumber daya ingress mengonfigurasi ALB untuk merutekan lalu lintas HTTP atau HTTPS ke Pod yang berbeda di dalam kluster. Untuk memastikan bahwa objek ingress Anda menggunakan AWS Load Balancer Controller, tambahkan anotasi berikut ke spesifikasi ingress Kubernetes Anda. Untuk informasi selengkapnya, lihat [spesifikasi Ingress](#) pada GitHub.

```
annotations:
  kubernetes.io/ingress.class: alb
```

Note

Jika Anda load balancing ke IPv6 Pod, tambahkan anotasi berikut ke spesifikasi ingress Anda. Anda hanya dapat memuat saldo IPv6 ke target IP, bukan target instance. Tanpa anotasi ini, load balancing selesai. IPv4

```
alb.ingress.kubernetes.io/ip-address-type: dualstack
```

- AWS Load Balancer Controller mendukung mode lalu lintas berikut:
 - Instans – Daftarkan simpul dalam kluster Anda sebagai target untuk ALB. Lalu lintas yang mencapai ALB dialihkan ke NodePort layanan Anda dan kemudian diproksi ke Pod Anda. Ini

adalah mode lalu lintas default. Anda juga dapat secara eksplisit menentukannya dengan anotasi `alb.ingress.kubernetes.io/target-type: instance`.

Note

Layanan Kubernetes Anda harus menentukan LoadBalancer jenis NodePort atau untuk menggunakan mode lalu lintas ini.

- IP — Mendaftarkan Pod sebagai target untuk ALB. Lalu lintas yang mencapai ALB langsung dirutekan ke Pod untuk layanan Anda. Anda harus menentukan anotasi `alb.ingress.kubernetes.io/target-type: ip` untuk menggunakan mode lalu lintas ini. Tipe target IP diperlukan saat Pod target berjalan di Fargate atau Amazon EKS Hybrid Nodes.
- Untuk tag yang ALBs dibuat oleh controller, tambahkan anotasi berikut ke `controller:alb.ingress.kubernetes.io/tags`. Untuk daftar semua anotasi yang tersedia yang didukung oleh AWS Load Balancer Controller, [lihat anotasi Ingress](#) pada GitHub
- Memutakhirkan atau menurunkan versi pengontrol ALB dapat memperkenalkan perubahan yang melanggar untuk fitur yang mengandalkannya. Untuk informasi selengkapnya tentang perubahan yang melanggar yang diperkenalkan di setiap rilis, lihat [catatan rilis pengontrol ALB](#) di GitHub.

Gunakan kembali ALBs dengan Grup Ingress

Anda dapat berbagi penyeimbang beban aplikasi di beberapa sumber daya layanan menggunakan `IngressGroups`.

Untuk menggabungkan ingress ke grup, tambahkan anotasi berikut ke spesifikasi sumber daya ingress Kubernetes.

```
alb.ingress.kubernetes.io/group.name: my-group
```

Nama grup harus:

- Panjangnya 63 atau lebih sedikit karakter.
- Terdiri dari huruf kecil, angka-, dan .
- Dimulai dan diakhiri dengan huruf atau angka.

Kontroler secara otomatis menggabungkan aturan masuk untuk semua ingress dalam grup ingress yang sama. Ini mendukung mereka dengan satu ALB. Sebagian besar anotasi yang didefinisikan

pada ingress hanya berlaku untuk jalur yang ditentukan oleh ingress tersebut. Secara default, sumber daya ingress bukan milik grup ingress mana pun.

Warning

Potensi risiko keamanan

Tentukan grup ingress untuk ingress hanya jika semua pengguna Kubernetes yang memiliki izin RBAC untuk membuat atau memodifikasi sumber daya ingress berada dalam batas kepercayaan yang sama. Jika Anda menambahkan anotasi dengan nama grup, pengguna Kubernetes lainnya mungkin membuat atau memodifikasi ingresses mereka untuk menjadi milik grup ingress yang sama. Melakukan hal tersebut dapat menyebabkan perilaku yang tidak diinginkan, seperti menimpa aturan yang ada dengan aturan prioritas yang lebih tinggi.

Anda dapat menambahkan nomor pesanan sumber daya masuk Anda.

```
alb.ingress.kubernetes.io/group.order: '10'
```

Jumlahnya bisa 1-1000. Angka terendah untuk semua ingress dalam kelompok ingress yang sama dievaluasi terlebih dahulu. Semua masuknya tanpa anotasi ini dievaluasi dengan nilai nol. Aturan duplikat angka yang lebih tinggi dapat menimpa aturan dengan angka yang lebih rendah. Secara default, urutan aturan antara ingress dalam grup ingress yang sama ditentukan namespace dan nama berbasis leksikografis.

Important

Pastikan bahwa setiap ingress dalam grup ingress yang sama memiliki nomor prioritas yang unik. Anda tidak dapat memiliki nomor urutan duplikat di seluruh ingresses.

(Opsional) Deploy aplikasi sampel

- Setidaknya satu subnet publik atau pribadi di VPC cluster Anda.
- Minta AWS Load Balancer Controller diterapkan di cluster Anda. Untuk informasi selengkapnya, lihat [the section called “AWS Pengontrol Load Balancer”](#). Kami merekomendasikan versi 2.7.2 atau yang lebih baru.

Anda dapat menjalankan aplikasi sampel pada kluster yang memiliki EC2 node Amazon, Pod Fargate, atau keduanya.

1. Jika Anda tidak menerapkan ke Fargate, lewati langkah ini. Jika Anda menerapkan ke Fargate, buat profil Fargate. Anda dapat membuat profil dengan menjalankan perintah berikut atau [AWS Management Console](#) menggunakan nilai yang sama untuk name dan namespace yang ada di perintah. Ganti *example values* dengan milik Anda sendiri.

```
eksctl create fargateprofile \  
  --cluster my-cluster \  
  --region region-code \  
  --name alb-sample-app \  
  --namespace game-2048
```

2. Terapkan game [2048](#) sebagai contoh aplikasi untuk memverifikasi bahwa Load AWS Balancer Controller membuat AWS ALB sebagai hasil dari objek ingress. Selesaikan langkah-langkah untuk jenis subnet yang Anda gunakan.

- a. Jika Anda menerapkan ke Pod dalam kluster yang Anda buat bersama IPv6 keluarga, lewati ke langkah berikutnya.

- Publik::

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.11.0/docs/examples/2048/2048_full.yaml
```

- Pribadi::

- A. Unduh manifes.

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.11.0/docs/examples/2048/2048_full.yaml
```

- B. Edit file dan temukan baris yang bertuliskan `alb.ingress.kubernetes.io/scheme: internet-facing`.

- C. Ubah *internet-facing* ke `internal` dan simpan file.

- D. Menerapkan manifes ke kluster Anda.

```
kubectl apply -f 2048_full.yaml
```


- b. Jika Anda menerapkan ke Pod dalam kluster yang Anda buat bersama [IPv6 keluarga](#), selesaikan langkah-langkah berikut.

- i. Unduh manifes.

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.11.0/docs/examples/2048/2048_full.yaml
```

- ii. Buka file di editor dan tambahkan baris berikut ke anotasi dalam spesifikasi ingress.

```
alb.ingress.kubernetes.io/ip-address-type: dualstack
```

- iii. Jika Anda load balancing ke Pod internal, bukan Pod yang menghadap ke internet, ubah baris yang mengatakan `alb.ingress.kubernetes.io/scheme: internet-facing` ke `alb.ingress.kubernetes.io/scheme: internal`

- iv. Simpan file tersebut.

- v. Menerapkan manifes ke kluster Anda.

```
kubectl apply -f 2048_full.yaml
```

3. Setelah beberapa menit, verifikasi bahwa sumber daya ingress dibuat dengan perintah berikut.

```
kubectl get ingress/ingress-2048 -n game-2048
```

Contoh output adalah sebagai berikut.

NAME	CLASS	HOSTS	ADDRESS
		PORTS	AGE
ingress-2048	<none>	*	k8s-game2048-ingress2-xxxxxxxxxx-yyyyyyyyyy.region-code.elb.amazonaws.com
		80	2m32s

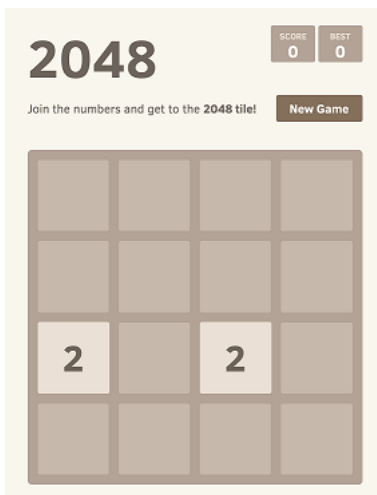
Note

Jika Anda membuat penyeimbang beban di subnet pribadi, nilai ADDRESS di bawah output sebelumnya diawali dengan. `internal-`

Jika ingress Anda tidak berhasil dibuat setelah beberapa menit, jalankan perintah berikut untuk melihat log AWS Load Balancer Controller. Log ini mungkin berisi pesan kesalahan yang dapat Anda gunakan untuk mendiagnosis masalah dengan penerapan Anda.

```
kubectl logs -f -n kube-system -l app.kubernetes.io/instance=aws-load-balancer-controller
```

1. Jika Anda menggunakan subnet publik, buka browser dan arahkan ke ADDRESS URL dari output perintah sebelumnya untuk melihat contoh aplikasi. Jika Anda tidak melihat apa pun, segarkan browser Anda dan coba lagi. Jika Anda menerapkan ke subnet pribadi, maka Anda harus melihat halaman dari perangkat dalam VPC Anda, seperti host bastion. Untuk informasi selengkapnya, lihat [Linux Bastion Host di AWS](#).



2. Ketika Anda selesai bereksperimen dengan aplikasi sampel Anda, hapus dengan menjalankan salah satu perintah berikut.
 - Jika Anda menerapkan manifes, daripada menerapkan salinan yang Anda unduh, gunakan perintah berikut.

```
kubectl delete -f https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.11.0/docs/examples/2048/2048_full.yaml
```

- Jika Anda mengunduh dan mengedit manifes, gunakan perintah berikut.

```
kubectl delete -f 2048_full.yaml
```

Batasi alamat IP eksternal yang dapat ditetapkan ke layanan

Layanan Kubernetes dapat dicapai dari dalam kluster melalui:

- Alamat IP kluster yang ditetapkan secara otomatis oleh Kubernetes
- Alamat IP apa pun yang Anda tentukan untuk properti `externalIPs` dalam spesifikasi layanan. Alamat IP eksternal tidak dikelola oleh Kubernetes dan merupakan tanggung jawab administrator kluster. Alamat IP eksternal yang ditentukan dengan `externalIPs` berbeda dari alamat IP eksternal yang ditetapkan ke layanan jenis LoadBalancer oleh penyedia cloud.

Untuk mempelajari selengkapnya tentang layanan Kubernetes, lihat [Layanan](#) dalam dokumentasi Kubernetes. Anda dapat membatasi alamat IP yang dapat ditentukan untuk `externalIPs` dalam spesifikasi layanan.

1. Terapkan `cert-manager` untuk mengelola sertifikat webhook. Untuk informasi selengkapnya, lihat dokumentasi [cert manager](#).

```
kubectl apply -f https://github.com/jetstack/cert-manager/releases/download/v1.5.4/cert-manager.yaml
```

2. Verifikasi bahwa `cert-manager` Pod sedang berjalan.

```
kubectl get pods -n cert-manager
```

Contoh output adalah sebagai berikut.

NAME	READY	STATUS	RESTARTS	AGE
cert-manager-58c8844bb8-nlx7q	1/1	Running	0	15s
cert-manager-cainjector-745768f6ff-696h5	1/1	Running	0	15s
cert-manager-webhook-67cc76975b-4v4nk	1/1	Running	0	14s

3. Tinjau layanan Anda yang ada untuk memastikan bahwa tidak satupun dari mereka memiliki alamat IP eksternal yang ditetapkan untuk mereka yang tidak terkandung dalam blok CIDR yang ingin Anda batasi alamatnya.

```
kubectl get services -A
```

Contoh output adalah sebagai berikut.

NAMESPACE	EXTERNAL-IP	NAME	TYPE
CLUSTER-IP		PORT(S)	AGE
cert-manager		cert-manager	ClusterIP
10.100.102.137	<none>	9402/TCP	20m
cert-manager		cert-manager-webhook	ClusterIP
10.100.6.136	<none>	443/TCP	20m
default		kubernetes	ClusterIP
10.100.0.1	<none>	443/TCP	2d1h
externalip-validation-system		externalip-validation-webhook-service	ClusterIP
10.100.234.179	<none>	443/TCP	16s
kube-system		kube-dns	ClusterIP
10.100.0.10	<none>	53/UDP,53/TCP	2d1h
my-namespace		my-service	ClusterIP
10.100.128.10	192.168.1.1	80/TCP	149m

Jika salah satu nilai adalah alamat IP yang tidak berada dalam blok yang ingin Anda batasi aksesnya, Anda harus mengubah alamat untuk berada di dalam blok, dan menerapkan kembali layanan. Misalnya, `my-service` layanan di output sebelumnya memiliki alamat IP eksternal yang ditetapkan untuk itu yang tidak berada dalam contoh blok CIDR di langkah 5.

- Unduh manifes webhook IP eksternal. Anda juga dapat melihat [kode sumber untuk webhook](#) di GitHub.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/docs/externalip-webhook.yaml
```

- Tentukan blok CIDR. Buka file yang diunduh dalam editor Anda dan hapus `\#` pada awal baris berikut.

```
#args:
#- --allowed-external-ip-cidrs=10.0.0.0/8
```

Ganti `10.0.0.0/8` dengan blok CIDR Anda sendiri. Anda dapat menentukan blok sebanyak yang Anda inginkan. Jika Anda menentukan beberapa blok, tambahkan koma di antara blok.

- Jika cluster Anda tidak berada di `us-west-2` AWS Region, maka ganti `us-west-2602401143452`, dan `amazonaws.com` dalam file dengan perintah berikut. Sebelum menjalankan perintah, ganti `region-code` dan `111122223333` dengan nilai untuk AWS Wilayah Anda dari daftar di [Lihat pendaftar gambar penampung Amazon untuk add-on Amazon EKS](#).

```
sed -i.bak -e 's|602401143452|111122223333|' externalip-webhook.yaml
```

```
sed -i.bak -e 's|us-west-2|region-code|' externalip-webhook.yaml
sed -i.bak -e 's|amazonaws.com||' externalip-webhook.yaml
```

7. Menerapkan manifes ke kluster Anda.

```
kubectl apply -f externalip-webhook.yaml
```

Upaya untuk menyebarkan layanan ke kluster Anda dengan alamat IP yang ditentukan untuk externalIPs itu tidak terkandung dalam blok yang Anda tentukan dalam langkah Tentukan blok CIDR akan gagal.

Salin gambar kontainer dari satu repositori ke repositori lain

Topik ini menjelaskan cara menarik gambar kontainer dari repositori yang tidak dapat diakses oleh node Anda dan mendorong gambar ke repositori yang dapat diakses oleh node Anda. Anda dapat mendorong gambar ke Amazon ECR atau repositori alternatif yang dapat diakses oleh node Anda.

- Mesin Docker diinstal dan dikonfigurasi di komputer Anda. Untuk petunjuk, lihat [Menginstal Docker Engine](#) di dokumentasi Docker.
- Versi 2.12.3 atau yang lebih baru atau versi 1.27.160 atau yang lebih baru dari AWS Command Line Interface (AWS CLI) diinstal dan dikonfigurasi pada perangkat Anda atau AWS CloudShell. Untuk memeriksa versi Anda saat ini, gunakan `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Package manager seperti `yum` atau `apt-get`, atau Homebrew untuk macOS seringkali merupakan beberapa versi di belakang versi terbaru CLI. AWS Untuk menginstal versi terbaru, lihat [Menginstal](#) dan [Konfigurasi cepat dengan aws configure](#) di Panduan Pengguna Antarmuka Baris Perintah. Versi AWS CLI yang diinstal AWS CloudShell mungkin juga beberapa versi di belakang versi terbaru. Untuk memperbaruinya, lihat [Menginstal AWS CLI ke direktori home Anda](#) di AWS CloudShell Panduan Pengguna.
- Titik akhir VPC antarmuka untuk Amazon ECR jika Anda ingin node Anda menarik gambar kontainer dari atau mendorong gambar kontainer ke repositori ECR Amazon pribadi melalui jaringan Amazon. Untuk informasi selengkapnya, lihat [Membuat titik akhir VPC untuk Amazon ECR di Panduan Pengguna Amazon Elastic Container Registry](#).

Selesaikan langkah-langkah berikut untuk menarik gambar kontainer dari repositori dan mendorongnya ke repositori Anda sendiri. Dalam contoh berikut yang disediakan dalam topik ini, gambar untuk [plugin Amazon VPC CNI untuk pembantu metrik Kubernetes ditarik](#). Ketika Anda

mengikuti langkah-langkah ini, pastikan untuk mengganti *example values* dengan nilai Anda sendiri.

1. Jika Anda belum memiliki repositori Amazon ECR atau repositori lain, maka buat repositori yang dapat diakses oleh node Anda. Perintah berikut membuat repositori pribadi Amazon ECR. Nama repositori pribadi Amazon ECR harus dimulai dengan huruf. Ini hanya dapat berisi huruf kecil, angka, tanda hubung (-), garis bawah (_), dan garis miring maju (/). Untuk informasi selengkapnya, lihat [Membuat repositori pribadi](#) di Panduan Pengguna Amazon Elastic Container Registry.

Anda dapat mengganti *cni-metrics-helper* dengan apa pun yang Anda pilih. Sebagai praktik terbaik, buat repositori terpisah untuk setiap gambar. Kami merekomendasikan ini karena tag gambar harus unik dalam repositori. Ganti *region-code* dengan [AWS Wilayah yang didukung oleh Amazon ECR](#).

```
aws ecr create-repository --region region-code --repository-name cni-metrics-helper
```

2. Tentukan registri, repositori, dan tag (opsional) dari gambar yang perlu ditarik oleh node Anda. Informasi ini dalam `registry/repository[:tag]` format.

Banyak topik Amazon EKS tentang menginstal gambar mengharuskan Anda menerapkan file manifes atau menginstal gambar menggunakan bagan Helm. Namun, sebelum Anda menerapkan file manifes atau menginstal bagan Helm, pertama-tama lihat konten manifes atau `values.yaml` file bagan. Dengan begitu, Anda dapat menentukan registri, repositori, dan tag yang akan ditarik.

Misalnya, Anda dapat menemukan baris berikut dalam [file manifes](#) untuk [plugin Amazon VPC CNL untuk pembantu metrik Kubernetes](#). Registri adalah `602401143452.dkr.ecr.us-west-2.amazonaws.com`, yang merupakan registri pribadi Amazon ECR. Repositori adalah `cni-metrics-helper`

```
image: "602401143452.dkr.ecr.us-west-2.amazonaws.com/cni-metrics-helper:v1.12.6"
```

Anda dapat melihat variasi berikut untuk lokasi gambar:

- `repository-name:tag`. Dalam hal ini, `docker.io` biasanya registri, tetapi tidak ditentukan karena Kubernetes menambahkannya ke nama repositori secara default jika tidak ada registri yang ditentukan.
- `repository-name/repository-namespace/repository:tag`. Namespace repositori bersifat opsional, tetapi terkadang ditentukan oleh pemilik repositori untuk mengkategorikan

gambar. Misalnya, semua [EC2 gambar Amazon di Galeri Publik Amazon ECR](#) menggunakan `aws-ec2` namespace.

Sebelum menginstal gambar dengan Helm, lihat `values.yaml` file Helm untuk menentukan lokasi gambar. Misalnya, file [values.yaml](#) untuk [plugin Amazon VPC CNI untuk pembantu metrik Kubernetes](#) menyertakan baris berikut.

```
image:
  region: us-west-2
  tag: v1.12.6
  account: "602401143452"
  domain: "amazonaws.com"
```

3. Tarik gambar kontainer yang ditentukan dalam file manifes.

- a. Jika Anda menarik dari registri publik, seperti [Galeri Publik Amazon ECR](#), Anda dapat melompat ke sub-langkah berikutnya, karena otentikasi tidak diperlukan. Dalam contoh ini, Anda mengautentikasi ke registri pribadi Amazon ECR yang berisi repositori untuk gambar helper metrik CNI. Amazon EKS mempertahankan gambar di setiap registri yang terdaftar di [Lihat pendaftar gambar kontainer Amazon untuk add-on Amazon EKS](#). Anda dapat mengautentikasi ke salah satu pendaftar dengan mengganti `602401143452` dan `region-code` dengan informasi untuk registri yang berbeda. Registri terpisah ada untuk setiap [AWS Wilayah tempat Amazon EKS didukung](#).

```
aws ecr get-login-password --region region-code | docker login --username AWS --password-stdin 602401143452.dkr.ecr.region-code.amazonaws.com
```

- b. Tarik gambar. Dalam contoh ini, Anda menarik dari registri yang Anda autentikasi di sub-langkah sebelumnya. Ganti `602401143452` dan `region-code` dengan informasi yang Anda berikan di sub-langkah sebelumnya.

```
docker pull 602401143452.dkr.ecr.region-code.amazonaws.com/cni-metrics-helper:v1.12.6
```

4. Tandai gambar yang Anda tarik dengan registri, repositori, dan tag Anda. Contoh berikut mengasumsikan bahwa Anda menarik gambar dari file manifes dan akan mendorongnya ke repositori pribadi Amazon ECR yang Anda buat pada langkah pertama. Ganti `111122223333` dengan ID akun Anda. Ganti `region-code` dengan AWS Wilayah tempat Anda membuat repositori pribadi Amazon ECR.

```
docker tag cni-metrics-helper:v1.12.6 111122223333.dkr.ecr.region-code.amazonaws.com/cni-metrics-helper:v1.12.6
```

5. Otentikasi ke registri Anda. Dalam contoh ini, Anda mengautentikasi ke registri pribadi Amazon ECR yang Anda buat pada langkah pertama. Untuk informasi selengkapnya, lihat [Autentikasi registri](#) di Panduan Pengguna Amazon Elastic Container Registry.

```
aws ecr get-login-password --region region-code | docker login --username AWS --password-stdin 111122223333.dkr.ecr.region-code.amazonaws.com
```

6. Dorong gambar ke repositori Anda. Dalam contoh ini, Anda mendorong gambar ke repositori pribadi Amazon ECR yang Anda buat pada langkah pertama. Untuk informasi selengkapnya, lihat [Mendorong gambar Docker](#) di Panduan Pengguna Amazon Elastic Container Registry.

```
docker push 111122223333.dkr.ecr.region-code.amazonaws.com/cni-metrics-helper:v1.12.6
```

7. Perbarui file manifes yang Anda gunakan untuk menentukan gambar pada langkah sebelumnya dengan gambar yang Anda dorong. `registry/repository:tag` Jika Anda menginstal dengan bagan Helm, sering ada opsi untuk menentukan. `registry/repository:tag` Saat memasang bagan, tentukan gambar `registry/repository:tag` yang Anda dorong ke repositori Anda.

Lihat pendaftar gambar kontainer Amazon untuk add-on Amazon EKS

Saat Anda menerapkan [add-on AWS Amazon EKS](#) ke kluster Anda, node Anda akan menarik gambar kontainer yang diperlukan dari registri yang ditentukan dalam mekanisme penginstalan untuk add-on, seperti manifes instalasi atau file Helm. `values.yaml` Gambar diambil dari repositori pribadi Amazon EKS Amazon ECR. Amazon EKS mereplikasi gambar ke repositori di setiap Wilayah yang didukung Amazon EKS. AWS Node Anda dapat menarik gambar kontainer melalui internet dari salah satu pendaftar berikut. Atau, node Anda dapat menarik gambar melalui jaringan Amazon jika Anda membuat titik [akhir VPC antarmuka untuk Amazon ECR \(AWS PrivateLink di VPC](#) Anda. Registri memerlukan otentikasi dengan akun AWS IAM. Node Anda mengautentikasi menggunakan [peran IAM node Amazon EKS](#), yang memiliki izin dalam kebijakan IAM `EC2 ContainerRegistryReadOnly` terkelola [Amazon](#) yang terkait dengannya.

AWS Wilayah	Registri
af-south-1	877085696533.dkr. ecr.af-south-1.amazonaws.com
ap-east-1	800184023465.dkr. ecr.ap-east-1.amazonaws.com
ap-timur-2	533267051163.dkr. ecr.ap-east-1.amazonaws.com
ap-southeast-3	296578399912.dkr. ecr.ap-southeast-3.amazonaws.com
ap-south-2	900889452093.dkr. ecr.ap-south-2.amazonaws.com
ap-southeast-4	491585149902.dkr. ecr.ap-southeast-4.amazonaws.com
ap-south-1	602401143452.dkr. ecr.ap-south-1.amazonaws.com
ap-northeast-3	602401143452.dkr. ecr.ap-northeast-3.amazonaws.com
ap-northeast-2	602401143452.dkr. ecr.ap-northeast-2.amazonaws.com
ap-southeast-1	602401143452.dkr. ecr.ap-southeast-1.amazonaws.com
ap-southeast-2	602401143452.dkr. ecr.ap-southeast-2.amazonaws.com
ap-tenggara 7	121268973566.dkr. ecr.ap-southeast-7.amazonaws.com

AWS Wilayah	Registri
ap-northeast-1	602401143452.dkr. ecr.ap-northeast-1. .amazonaws.com
cn-north-1	918309763551.dkr. ecr.cn-north-1.ama zonaws.com .cn
cn-northwest-1	961992271922.dkr. ecr.cn-northwest-1 .amazonaws.com .cn
eu-central-1	602401143452.dkr. ecr.eu-central-1.a mazonaws.com
eu-west-1	602401143452.dkr. ecr.eu-west-1.amaz onaws.com
eu-west-2	602401143452.dkr. ecr.eu-west-2.amaz onaws.com
eu-south-1	590381155156.dkr. ecr.eu-south-1.ama zonaws.com
eu-west-3	602401143452.dkr. ecr.eu-west-3.amaz onaws.com
eu-south-2	455263428931.dkr. ecr.eu-south-2.ama zonaws.com
eu-north-1	602401143452.dkr. ecr.eu-north-1.ama zonaws.com
eu-central-2	900612956339.dkr. ecr.eu-central-2.a mazonaws.com
il-central-1	066635153087.dkr. ecr.il-central-1.amazonaws. com
mx-pusat-1	730335286997.dkr. ecr.mx-central-1.a mazonaws.com

AWS Wilayah	Registri
me-south-1	558608220178.dkr. ecr.me-south-1.amazonaws.com
me-central-1	759879836304.dkr. ecr.me-central-1.amazonaws.com
us-east-1	602401143452.dkr. ecr.us-east-1.amazonaws.com
us-east-2	602401143452.dkr. ecr.us-east-2.amazonaws.com
us-west-1	602401143452.dkr. ecr.us-west-1.amazonaws.com
us-west-2	602401143452.dkr. ecr.us-west-2.amazonaws.com
ca-central-1	602401143452.dkr. ecr.ca-central-1.amazonaws.com
ca-west-1	761377655185.dkr. ecr.ca-west-1.amazonaws.com
sa-east-1	602401143452.dkr. ecr.sa-east-1.amazonaws.com
us-gov-east-1	151742754352.dkr. ecr. us-gov-east-1.amazonaws.com
us-gov-west-1	013241004608.dkr. ecr. us-gov-west-1.amazonaws.com

Add-on Amazon EKS

Add-on adalah perangkat lunak yang menyediakan kemampuan operasional pendukung untuk aplikasi Kubernetes, tetapi tidak spesifik kepada aplikasi. Ini termasuk perangkat lunak seperti agen

observabilitas atau driver Kubernetes yang memungkinkan cluster berinteraksi dengan AWS sumber daya yang mendasarinya untuk jaringan, komputasi, dan penyimpanan. Add-on software biasanya dibangun dan dikelola oleh komunitas Kubernetes, penyedia cloud seperti AWS, atau vendor pihak ketiga. Amazon EKS secara otomatis menginstal add-on yang dikelola sendiri seperti plugin Amazon VPC CNI untuk Kubernetes, dan CoreDNS untuk setiap cluster. kube-proxy Perhatikan bahwa add-on VPC CNI tidak kompatibel dengan Amazon EKS Hybrid Nodes dan tidak diterapkan ke node hybrid. Anda dapat mengubah konfigurasi default add-on dan memperbaruinya bila diinginkan.

Add-on Amazon EKS menyediakan instalasi dan pengelolaan serangkaian add-on yang dikuratori untuk kluster Amazon EKS. Semua add-on Amazon EKS menyertakan patch keamanan terbaru, perbaikan bug, dan divalidasi oleh untuk AWS bekerja dengan Amazon EKS. Add-on Amazon EKS memungkinkan Anda untuk secara konsisten memastikan bahwa kluster Amazon EKS Anda aman dan stabil serta mengurangi jumlah pekerjaan yang perlu Anda lakukan untuk menginstal, mengonfigurasi, dan memperbarui add-on. Jika add-on yang dikelola sendiri, seperti kube-proxy sudah berjalan di cluster Anda dan tersedia sebagai add-on Amazon EKS, maka Anda dapat menginstal add-on kube-proxy Amazon EKS untuk mulai memanfaatkan kemampuan add-on Amazon EKS.

Anda dapat memperbarui bidang konfigurasi terkelola Amazon EKS tertentu untuk add-on Amazon EKS melalui Amazon EKS API. Anda juga dapat memodifikasi bidang konfigurasi yang tidak dikelola oleh Amazon EKS langsung dalam kluster Kubernetes setelah add-on dimulai. Ini termasuk mendefinisikan bidang konfigurasi tertentu untuk add-on jika berlaku. Perubahan ini tidak ditimpa oleh Amazon EKS setelah mereka dibuat. Hal ini dimungkinkan dengan menggunakan fitur penerapan sisi server Kubernetes. Untuk informasi selengkapnya, lihat [the section called “Bidang yang dapat Anda sesuaikan”](#).

Anda dapat menggunakan add-on Amazon EKS dengan jenis node Amazon EKS apa pun. Untuk informasi selengkapnya, lihat [Kelola komputasi](#).

Anda dapat menambahkan, memperbarui, atau menghapus add-on Amazon EKS menggunakan Amazon EKS API,, AWS CLI AWS Management Console, dan. eksctl Anda juga dapat membuat Amazon EKS add-on menggunakan [AWS CloudFormation](#).

Pertimbangan

Pertimbangkan hal berikut saat Anda menggunakan add-on Amazon EKS:

- Untuk mengonfigurasi add-on untuk cluster, [prinsipal IAM](#) Anda harus memiliki izin IAM untuk bekerja dengan add-on. Untuk informasi selengkapnya, lihat tindakan dengan namanya Addon dalam [Tindakan yang ditentukan oleh Amazon Elastic Kubernetes Service](#).
- Add-on Amazon EKS berjalan pada simpul yang Anda sediakan atau konfigurasikan untuk kluster Anda. Jenis node termasuk EC2 instans Amazon, Fargate, dan node hybrid.
- Anda dapat memodifikasi bidang yang tidak dikelola oleh Amazon EKS untuk menyesuaikan penginstalan add-on Amazon EKS. Untuk informasi selengkapnya, lihat [the section called “Bidang yang dapat Anda sesuaikan”](#).
- Jika Anda membuat cluster dengan, Amazon EKS AWS Management Consolekubernetes-proxy, plugin Amazon VPC CNI untuk Kubernetes, dan add-on CoreDNS Amazon EKS secara otomatis ditambahkan ke cluster Anda. Jika Anda gunakan eksctl untuk membuat cluster Anda dengan config file, juga eksctl dapat membuat cluster dengan Amazon EKS add-on. Jika Anda membuat kluster Anda menggunakan eksctl tanpa config file atau dengan alat lain, plugin CNI Amazon VPC yang dikelola sendiri untuk Kuberneteskubernetes-proxy, dan add-on CoreDNS diinstal, bukan add-on Amazon EKS. Anda dapat mengelolanya sendiri atau menambahkan add-on Amazon EKS secara manual setelah pembuatan cluster. Terlepas dari metode yang Anda gunakan untuk membuat cluster Anda, add-on VPC CNI tidak diinstal pada node hybrid.
- Itu eks:addon-cluster-admin ClusterRoleBinding mengikat identitas cluster-admin ClusterRole eks:addon-manager Kubernetes. Peran tersebut memiliki izin yang diperlukan untuk eks:addon-manager identitas untuk membuat ruang nama Kubernetes dan menginstal add-on ke dalam ruang nama. Jika eks:addon-cluster-admin ClusterRoleBinding dihapus, kluster Amazon EKS akan terus berfungsi, namun Amazon EKS tidak lagi dapat mengelola add-on apa pun. Semua cluster yang dimulai dengan versi platform berikut menggunakan yang baruClusterRoleBinding.
- Subset dari add-on EKS dari AWS telah divalidasi untuk kompatibilitas dengan Amazon EKS Hybrid Nodes. Untuk informasi selengkapnya, lihat tabel kompatibilitas di[the section called “ AWS add-on”](#).

Pertimbangan untuk Mode Otomatis Amazon EKS

Mode Amazon EKS Auto mencakup kemampuan yang menghadirkan fungsionalitas kluster penting, termasuk:

- Jaringan Pod
- Jaringan layanan

- Kluster DNS
- Penskalaan otomatis
- Penyimpanan blok
- Pengontrol penyeimbang beban
- Agen Pod Identity
- Agen pemantauan simpul

Dengan komputasi mode Otomatis, banyak add-on EKS yang umum digunakan menjadi berlebihan, seperti:

- Amazon VPC CNI
- kube-proxy
- CoreDNS
- Pengemudi Amazon EBS CSI
- Agen Identitas EKS Pod

Namun, jika klaster Anda menggabungkan mode Otomatis dengan opsi komputasi lainnya seperti EC2 instans yang dikelola sendiri, Grup Node Terkelola, atau AWS Fargate, add-on ini tetap diperlukan. AWS telah menyempurnakan add-on EKS dengan aturan anti-afinitas yang secara otomatis memastikan pod add-on dijadwalkan hanya pada jenis komputasi yang didukung. Selain itu, pengguna sekarang dapat memanfaatkan `DescribeAddonVersions` API add-on EKS untuk memverifikasi `ComputeTypes` yang didukung untuk setiap add-on dan versi spesifiknya. Selain itu, dengan mode EKS Auto, pengontrol yang tercantum di atas berjalan pada infrastruktur yang AWS dimiliki. Jadi, Anda banyak bahkan tidak melihatnya di akun Anda kecuali Anda menggunakan mode otomatis EKS dengan jenis komputasi lain dalam hal ini, Anda akan melihat pengontrol yang Anda instal di cluster Anda.

Jika Anda berencana untuk mengaktifkan Mode Otomatis EKS pada cluster yang ada, Anda mungkin perlu memutakhirkan versi addon tertentu. Untuk informasi selengkapnya, lihat [the section called “Versi add-on yang diperlukan”](#) untuk Mode Otomatis EKS.

Dukungan

AWS menerbitkan beberapa jenis add-on dengan berbagai tingkat dukungan.

- AWS Add-on: Add-on ini dibuat dan didukung sepenuhnya oleh AWS

- Gunakan AWS add-on untuk bekerja dengan AWS layanan lain, seperti Amazon EFS.
- Untuk informasi selengkapnya, lihat [the section called “AWS add-on”](#).
- AWS Marketplace Add-on: Add-on ini dipindai oleh AWS dan didukung oleh mitra independen.
AWS
 - Gunakan add-on marketplace untuk menambahkan fitur berharga dan canggih ke cluster Anda, seperti pemantauan dengan Splunk.
 - Untuk informasi selengkapnya, lihat [the section called “Pengaya Marketplace”](#).
- Add-on Komunitas: Add-on ini dipindai oleh AWS tetapi didukung oleh komunitas open source.
 - Gunakan add-on komunitas untuk mengurangi kerumitan menginstal perangkat lunak open source umum, seperti Kubernetes Metrics Server.
 - Add-on komunitas dibangun oleh AWS. AWS hanya memvalidasi add-on komunitas untuk kompatibilitas versi.
 - Untuk informasi selengkapnya, lihat [the section called “Pengaya komunitas”](#).

Tabel berikut merinci ruang lingkup dukungan untuk setiap jenis add-on:

Kategori	Fitur	AWS add-on	AWS Pengaya Marketplace	Pengaya komunitas
Pengembangan	Dibangun oleh AWS	Ya	Tidak	Ya
Pengembangan	Divalidasi oleh AWS	Ya	Tidak	Ya*
Pengembangan	Divalidasi oleh Mitra AWS	Tidak	Ya	Tidak
Maintenance	Dipindai oleh AWS	Ya	Ya	Ya
Maintenance	Ditambah oleh AWS	Ya	Tidak	Ya
Maintenance	Ditambah oleh Mitra AWS	Tidak	Ya	Tidak

Kategori	Fitur	AWS add-on	AWS Pengaya Marketplace	Pengaya komunitas
Distribusi	Diterbitkan oleh AWS	Ya	Tidak	Ya
Distribusi	Diterbitkan oleh AWS Partner	Tidak	Ya	Tidak
Dukungan	Dukungan Instal Dasar oleh AWS	Ya	Ya	Ya
Dukungan	AWS Support Penuh	Ya	Tidak	Tidak
Dukungan	Dukungan AWS Partner Penuh	Tidak	Ya	Tidak

*: Validasi untuk add-on komunitas hanya mencakup kompatibilitas versi Kubernetes. Misalnya, jika Anda menginstal add-on komunitas di sebuah klaster, AWS periksa apakah itu kompatibel dengan versi Kubernetes dari klaster Anda.

AWS Add-on Marketplace dapat mengunduh dependensi perangkat lunak tambahan dari sumber eksternal di luar. AWS Dependensi eksternal ini tidak dipindai atau divalidasi oleh. AWS Pertimbangkan persyaratan keamanan Anda saat menerapkan add-on AWS Marketplace yang mengambil dependensi eksternal.

AWS add-on

Add-on Amazon EKS berikut tersedia untuk dibuat di cluster Anda. Anda dapat melihat daftar terbaru dari add-on yang tersedia menggunakan `eksctl`, AWS Management Console, atau AWS CLI. Untuk melihat semua add-on yang tersedia atau untuk menginstal add-on, lihat [the section called “Buat add-on”](#). Jika add-on memerlukan izin IAM, maka Anda harus memiliki penyedia IAM OpenID Connect (OIDC) untuk klaster Anda. Untuk menentukan apakah Anda memilikinya, atau membuatnya, lihat [the section called “Penyedia IAM OIDC”](#). Anda dapat membuat atau menghapus add-on setelah Anda menginstalnya. Untuk informasi selengkapnya, lihat [the section called “Perbarui add-on”](#) atau [the section called “Hapus add-on”](#). Untuk informasi selengkapnya tentang pertimbangan

husus untuk menjalankan add-on EKS dengan Amazon EKS Hybrid Nodes, lihat. [the section called “Konfigurasi add-on”](#)

Anda dapat menggunakan salah satu add-on Amazon EKS berikut.

Deskripsi	Pelajari selengkapnya	Jenis komputasi yang kompatibel
Menyediakan jaringan VPC asli untuk cluster Anda	the section called “Plugin Amazon VPC CNI untuk Kubernetes”	EC2
Server DNS yang fleksibel dan dapat diperluas yang dapat berfungsi sebagai DNS cluster Kubernetes	the section called “CoreDNS”	EC2, Fargate, Mode Otomatis EKS, Node Hibrida EKS
Pertahankan aturan jaringan di setiap EC2 node Amazon	the section called “Kube-proxy ”	EC2, Node Hibrida EKS
Menyediakan penyimpanan Amazon EBS untuk klaster Anda	the section called “Driver CSI Amazon EBS”	EC2
Menyediakan penyimpanan Amazon EFS untuk klaster Anda	the section called “Driver Amazon EFS CSI”	EC2, Mode Otomatis EKS
Menyediakan Amazon FSx untuk penyimpanan Lustre untuk klaster Anda	the section called “Pengemudi Amazon FSx CSI”	EC2, Mode Otomatis EKS
Menyediakan penyimpanan Amazon S3 untuk klaster Anda	the section called “Mountpoint untuk Driver Amazon S3 CSI”	EC2, Mode Otomatis EKS
Mendeteksi masalah kesehatan node tambahan	the section called “Agen pemantauan simpul”	EC2, Node Hibrida EKS

Deskripsi	Pelajari selengkapnya	Jenis komputasi yang kompatibel
Aktifkan penggunaan fungsionalitas snapshot di driver CSI yang kompatibel, seperti driver Amazon EBS CSI	the section called “Pengontrol snapshot CSI”	EC2, Fargate, Mode Otomatis EKS, Node Hibrida EKS
SageMaker HyperPod Tata kelola tugas mengoptimalkan alokasi dan penggunaan sumber daya komputasi di seluruh tim di kluster Amazon EKS, mengatasi inefisiensi dalam prioritas tugas dan berbagi sumber daya.	the section called “SageMaker HyperPod Tata kelola tugas Amazon”	EC2, Mode Otomatis EKS,
Amazon SageMaker HyperPod Observability AddOn menyediakan kemampuan pemantauan dan observabilitas yang komprehensif untuk HyperPod cluster.	the section called “Pengaya SageMaker HyperPod Observabilitas Amazon”	EC2, Mode Otomatis EKS,
Agan Kubernetes yang mengumpulkan dan melaporkan data aliran jaringan ke Amazon CloudWatch, memungkinkan pemantauan komprehensif koneksi TCP di seluruh node cluster.	the section called “AWS Agen Monitor Aliran Jaringan”	EC2, Mode Otomatis EKS

Deskripsi	Pelajari selengkapnya	Jenis komputasi yang kompatibel
Distribusi proyek yang aman, siap produksi, dan AWS didukung OpenTelemetry	the section called “AWS Distro untuk OpenTelemetry”	EC2, Fargate, Mode Otomatis EKS, Node Hibrida EKS
Layanan pemantauan keamanan yang menganalisis dan memproses sumber data dasar termasuk peristiwa AWS CloudTrail manajemen dan log aliran VPC Amazon. Amazon GuardDuty juga memproses fitur, seperti log audit Kubernetes dan pemantauan runtime	the section called “GuardDuty Agen Amazon”	EC2, Mode Otomatis EKS
Layanan pemantauan dan observabilitas yang disediakan oleh AWS. Add-on ini menginstal CloudWatch Agen dan mengaktifkan Sinyal CloudWatch Aplikasi dan CloudWatch Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS	the section called “Agen CloudWatch Observabilitas Amazon”	EC2, Mode Otomatis EKS, Node Hibrida EKS
Kemampuan untuk mengelola kredensial untuk aplikasi Anda, mirip dengan cara profil EC2 instans menyediakan kredensial ke instance EC2	the section called “Agen Identitas EKS Pod”	EC2, Node Hibrida EKS

Deskripsi	Pelajari selengkapnya	Jenis komputasi yang kompatibel
Aktifkan cert-manager untuk mengeluarkan sertifikat X.509 dari Private CA. AWS membutuhkan cert-manager.	the section called “AWS Konektor CA Pribadi untuk Kubernetes”	EC2, Fargate, Mode Otomatis EKS, Node Hibrida EKS

Plugin Amazon VPC CNI untuk Kubernetes

Plugin Amazon VPC CNI untuk add-on Kubernetes Amazon EKS adalah plugin Kubernetes Container Network Interface (CNI) yang menyediakan jaringan VPC asli untuk kluster Anda. Jenis add-on yang dikelola sendiri atau dikelola ini diinstal pada setiap EC2 node Amazon, secara default. Untuk informasi selengkapnya, lihat Plugin [Kubernetes Container Network Interface \(CNI\)](#).

Note

Anda tidak perlu menginstal add-on ini di kluster Amazon EKS Auto Mode. Untuk informasi selengkapnya, lihat [the section called “Pertimbangan untuk Mode Otomatis Amazon EKS”](#).

Nama add-on Amazon EKS adalah `vpc-cni`.

Izin IAM yang diperlukan

Add-on ini menggunakan peran IAM untuk kemampuan akun layanan Amazon EKS. Untuk informasi selengkapnya, lihat [the section called “Kredensial dengan IRSA”](#).

Jika kluster Anda menggunakan IPv4 keluarga, izin di [AmazonEks_CNI_Policy](#) diperlukan. [Jika kluster Anda menggunakan IPv6 keluarga, Anda harus membuat kebijakan IAM dengan izin dalam IPv6 mode.](#) Anda dapat membuat peran IAM, melampirkan salah satu kebijakan ke dalamnya, dan membuat anotasi akun layanan Kubernetes yang digunakan oleh add-on dengan perintah berikut.

Ganti `my-cluster` dengan nama cluster Anda dan `AmazonEKSVPCCNIRole` dengan nama untuk peran Anda. Jika kluster Anda menggunakan IPv6 keluarga, ganti `AmazonEKS_CNI_Policy` dengan nama kebijakan yang Anda buat. Perintah ini mengharuskan Anda menginstal [eksctl](#) di perangkat Anda. Jika Anda perlu menggunakan alat lain untuk membuat peran, lampirkan kebijakan

padanya, dan beri anotasi pada akun layanan Kubernetes, lihat. [the section called “Tetapkan peran IAM”](#)

```
eksctl create iamserviceaccount --name aws-node --namespace kube-system --cluster my-cluster --role-name AmazonEKSVPCCNIRole \
    --role-only --attach-policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy --approve
```

Perbarui informasi

Anda hanya dapat memperbarui satu versi minor dalam satu waktu. Misalnya, jika versi Anda saat ini 1.28.x-eksbuild.y dan Anda ingin memperbaruinya 1.30.x-eksbuild.y, maka Anda harus memperbarui versi Anda saat ini 1.29.x-eksbuild.y dan kemudian memperbaruinya lagi ke 1.30.x-eksbuild.y. Untuk informasi selengkapnya tentang memperbarui add-on, lihat [the section called “Update \(EKS add-on\)”](#).

CoreDNS

Add-on CoreDNS Amazon EKS adalah server DNS yang fleksibel dan dapat diperluas yang dapat berfungsi sebagai DNS cluster Kubernetes. Jenis pengaya ini yang dikelola sendiri atau dikelola telah diinstal, secara default, saat Anda membuat kluster. Saat Anda meluncurkan kluster Amazon EKS dengan setidaknya satu simpul, dua replika citra CoreDNS di-deploy secara default, terlepas dari jumlah simpul yang di-deploy di kluster Anda. Pod-pod CoreDNS menyediakan resolusi nama untuk semua Pod di dalam kluster. Anda dapat menerapkan Pod CoreDNS ke node Fargate jika kluster Anda menyertakan profil Fargate dengan namespace yang cocok dengan namespace untuk penerapan CoreDNS. Untuk informasi selengkapnya, lihat [the section called “Tentukan profil”](#)

Note

Anda tidak perlu menginstal add-on ini di kluster Amazon EKS Auto Mode. Untuk informasi selengkapnya, lihat [the section called “Pertimbangan untuk Mode Otomatis Amazon EKS”](#).

Nama add-on Amazon EKS adalah `coredns`.

Izin IAM yang diperlukan

Add-on ini tidak memerlukan izin apa pun.

Informasi tambahan

Untuk mempelajari lebih lanjut tentang CoreDNS, lihat [Menggunakan CoreDNS for Service Discovery dan Menyesuaikan Layanan DNS](#) dalam dokumentasi Kubernetes.

Kube-proxy

Add-on Kube-proxy Amazon EKS mempertahankan aturan jaringan pada setiap EC2 node Amazon. Ini memungkinkan komunikasi jaringan ke Pod Anda. Jenis add-on yang dikelola sendiri atau dikelola ini diinstal pada setiap EC2 node Amazon di kluster Anda, secara default.

Note

Anda tidak perlu menginstal add-on ini di kluster Amazon EKS Auto Mode. Untuk informasi selengkapnya, lihat [the section called “Pertimbangan untuk Mode Otomatis Amazon EKS”](#).

Nama add-on Amazon EKS adalah kube-proxy.

Izin IAM yang diperlukan

Add-on ini tidak memerlukan izin apa pun.

Perbarui informasi

Sebelum memperbarui versi Anda saat ini, pertimbangkan persyaratan berikut:

- Kube-proxy pada cluster Amazon EKS memiliki [kompatibilitas dan kebijakan miring](#) yang sama dengan Kubernetes.

Informasi tambahan

Untuk mempelajari selengkapnya kube-proxy, lihat [kube-proxy di dokumentasi Kubernetes](#).

Driver CSI Amazon EBS

Driver Amazon EBS CSI Amazon EKS add-on adalah plugin Kubernetes Container Storage Interface (CSI) yang menyediakan penyimpanan Amazon EBS untuk cluster Anda.

 Note

Anda tidak perlu menginstal add-on ini di kluster Amazon EKS Auto Mode. Mode Otomatis mencakup kemampuan penyimpanan blok. Untuk informasi selengkapnya, lihat [the section called “Menyebarkan beban kerja stateful”](#).

Nama add-on Amazon EKS adalah `aws-ebs-csi-driver`.

Izin IAM yang diperlukan

Add-on ini menggunakan peran IAM untuk kemampuan akun layanan Amazon EKS. Untuk informasi selengkapnya, lihat [the section called “Kredensial dengan IRSA”](#). Izin dalam kebijakan AWS terkelola [EBSCSIDriverKebijakan Amazon](#) diperlukan. Anda dapat membuat peran IAM dan melampirkan kebijakan terkelola padanya dengan perintah berikut. Ganti *my-cluster* dengan nama cluster Anda dan *AmazonEKS_EBS_CSI_DriverRole* dengan nama untuk peran Anda. Perintah ini mengharuskan Anda menginstal [eksctl](#) di perangkat Anda. Jika Anda perlu menggunakan alat yang berbeda atau Anda perlu menggunakan [kunci KMS](#) khusus untuk enkripsi, lihat [the section called “Langkah 1: Buat peran IAM”](#).

```
eksctl create iamserviceaccount \
  --name ebs-csi-controller-sa \
  --namespace kube-system \
  --cluster my-cluster \
  --role-name AmazonEKS_EBS_CSI_DriverRole \
  --role-only \
  --attach-policy-arn arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy \
  --approve
```

Informasi tambahan

Untuk mempelajari lebih lanjut tentang add-on, lihat [the section called “Amazon EBS”](#).

Driver Amazon EFS CSI

Add-on Amazon EFS CSI driver Amazon EKS adalah plugin Kubernetes Container Storage Interface (CSI) yang menyediakan penyimpanan Amazon EFS untuk cluster Anda.

Nama add-on Amazon EKS adalah `aws-efs-csi-driver`.

Izin IAM yang diperlukan

Izin IAM yang diperlukan - Pengaya ini menggunakan peran IAM untuk kemampuan akun layanan Amazon EKS. Untuk informasi selengkapnya, lihat [the section called “Kredensil dengan IRSA”](#). Izin dalam kebijakan AWS terkelola [EFSCSIDriverKebijakan Amazon](#) diperlukan. Anda dapat membuat peran IAM dan melampirkan kebijakan terkelola padanya dengan perintah berikut. Ganti *my-cluster* dengan nama cluster Anda dan *AmazonEKS_EFS_CSI_DriverRole* dengan nama untuk peran Anda. Perintah ini mengharuskan Anda menginstal [eksctl](#) di perangkat Anda. Jika Anda perlu menggunakan alat yang berbeda, lihat [the section called “Langkah 1: Buat peran IAM”](#).

```
export cluster_name=my-cluster
export role_name=AmazonEKS_EFS_CSI_DriverRole
eksctl create iamserviceaccount \
  --name efs-csi-controller-sa \
  --namespace kube-system \
  --cluster $cluster_name \
  --role-name $role_name \
  --role-only \
  --attach-policy-arn arn:aws:iam::aws:policy/service-role/AmazonEFSCSIDriverPolicy \
  --approve
TRUST_POLICY=$(aws iam get-role --output json --role-name $role_name --query
  'Role.AssumeRolePolicyDocument' | \
  sed -e 's/efs-csi-controller-sa/efs-csi-*/' -e 's/StringEquals/StringLike/')
aws iam update-assume-role-policy --role-name $role_name --policy-document
"$TRUST_POLICY"
```

Informasi tambahan

Untuk mempelajari lebih lanjut tentang add-on, lihat [the section called “Amazon EFS”](#).

Pengemudi Amazon FSx CSI

Add-on Amazon FSx CSI driver Amazon EKS adalah plugin Kubernetes Container Storage Interface (CSI) yang menyediakan Amazon FSx untuk penyimpanan Lustre untuk cluster Anda.

Nama add-on Amazon EKS adalah `aws-fsx-csi-driver`.

 Note

- Instalasi driver Amazon FSx CSI yang sudah ada sebelumnya di cluster dapat menyebabkan kegagalan penginstalan add-on. Saat Anda mencoba menginstal versi add-on Amazon EKS saat Driver FSx CSI non-EKS ada, penginstalan akan gagal karena konflik sumber daya. Gunakan OVERWRITE bendera selama instalasi untuk mengatasi masalah ini:

```
aws eks create-addon --addon-name aws-fsx-csi-driver --cluster-name my-cluster
--resolve-conflicts OVERWRITE
```

- Add-on Amazon FSx CSI Driver EKS memerlukan agen EKS Pod Identity untuk otentikasi. Tanpa komponen ini, add-on akan gagal dengan kesalahan `Amazon EKS Pod Identity agent is not installed in the cluster`, mencegah operasi volume. Instal agen Pod Identity sebelum atau sesudah menerapkan add-on Driver FSx CSI. Untuk informasi selengkapnya, lihat [the section called “Mengatur Agen”](#).

Izin IAM yang diperlukan

Add-on ini menggunakan peran IAM untuk kemampuan akun layanan Amazon EKS. Untuk informasi selengkapnya, lihat [the section called “Kredensial dengan IRSA”](#). Izin dalam kebijakan FSx FullAccess AWS dikelola [Amazon](#) diperlukan. Anda dapat membuat peran IAM dan melampirkan kebijakan dikelola padanya dengan perintah berikut. Ganti *my-cluster* dengan nama cluster Anda dan *AmazonEKS_FSx_CSI_DriverRole* dengan nama untuk peran Anda. Perintah ini mengharuskan Anda menginstal [eksctl](#) di perangkat Anda.

```
eksctl create iamserviceaccount \
  --name fsx-csi-controller-sa \
  --namespace kube-system \
  --cluster my-cluster \
  --role-name AmazonEKS_FSx_CSI_DriverRole \
  --role-only \
  --attach-policy-arn arn:aws:iam::aws:policy/AmazonFSxFullAccess \
  --approve
```

Informasi tambahan

Untuk mempelajari lebih lanjut tentang add-on, lihat [the section called “Amazon FSx untuk Lustre”](#).

Mountpoint untuk Driver Amazon S3 CSI

Mountpoint for Amazon S3 CSI Driver Amazon EKS add-on adalah plugin Kubernetes Container Storage Interface (CSI) yang menyediakan penyimpanan Amazon S3 untuk cluster Anda.

Nama add-on Amazon EKS adalah `aws-mountpoint-s3-csi-driver`.

Izin IAM yang diperlukan

Add-on ini menggunakan peran IAM untuk kemampuan akun layanan Amazon EKS. Untuk informasi selengkapnya, lihat [the section called “Kredensial dengan IRSA”](#).

Peran IAM yang dibuat akan memerlukan kebijakan yang memberikan akses ke S3. Ikuti [rekomendasi izin IAM Mountpoint](#) saat membuat kebijakan. Atau, Anda dapat menggunakan tautan kebijakan AWS terkelola: `iam/home?#/policies/arn:aws:iam:::aws:policy/AmazonS3 FullAccess` \$JSONEditor [AmazonS3, type="console"], tetapi kebijakan terkelola ini memberikan lebih banyak izin daripada yang diperlukan untuk MountpointFullAccess.

Anda dapat membuat peran IAM dan melampirkan kebijakan Anda padanya dengan perintah berikut. Ganti *my-cluster* dengan nama cluster Anda, *region-code* dengan kode AWS Region yang benar, *AmazonEKS_S3_CSI_DriverRole* dengan nama untuk peran Anda, dan *AmazonEKS_S3_CSI_DriverRole_ARN* dengan peran ARN. Perintah ini mengharuskan Anda menginstal [eksctl](#) di perangkat Anda. Untuk petunjuk tentang penggunaan konsol IAM atau AWS CLI, lihat [the section called “Langkah 2: Buat peran IAM”](#)

```
CLUSTER_NAME=my-cluster
REGION=region-code
ROLE_NAME=AmazonEKS_S3_CSI_DriverRole
POLICY_ARN=AmazonEKS_S3_CSI_DriverRole_ARN
eksctl create iamserviceaccount \
  --name s3-csi-driver-sa \
  --namespace kube-system \
  --cluster $CLUSTER_NAME \
  --attach-policy-arn $POLICY_ARN \
  --approve \
  --role-name $ROLE_NAME \
  --region $REGION \
  --role-only
```

Informasi tambahan

Untuk mempelajari lebih lanjut tentang add-on, lihat [the section called “Mountpoint untuk Amazon S3”](#).

Pengontrol snapshot CSI

Pengontrol snapshot Container Storage Interface (CSI) memungkinkan penggunaan fungsionalitas snapshot pada driver CSI yang kompatibel, seperti driver Amazon EBS CSI.

Nama add-on Amazon EKS adalah `snapshot-controller`.

Izin IAM yang diperlukan

Add-on ini tidak memerlukan izin apa pun.

Informasi tambahan

Untuk mempelajari lebih lanjut tentang add-on, lihat [the section called “Pengontrol snapshot CSI”](#).

SageMaker HyperPod Tata kelola tugas Amazon

SageMaker HyperPod Task Governance adalah sistem manajemen yang kuat yang dirancang untuk merampingkan alokasi sumber daya dan memastikan pemanfaatan sumber daya komputasi yang efisien di seluruh tim dan proyek untuk kluster Amazon EKS Anda. Ini memberi administrator kemampuan untuk mengatur:

- Tingkat prioritas untuk berbagai tugas
- Hitung alokasi untuk setiap tim
- Bagaimana setiap tim meminjamkan dan meminjam komputasi idle
- Jika tim mendahului tugas mereka sendiri

HyperPod Task governance juga menyediakan Observabilitas kluster Amazon EKS, menawarkan visibilitas real-time ke dalam kapasitas cluster. Ini termasuk ketersediaan dan penggunaan komputasi, alokasi dan pemanfaatan tim, dan informasi tugas dan waktu tunggu, menyiapkan Anda untuk pengambilan keputusan yang tepat dan manajemen sumber daya yang proaktif.

Nama add-on Amazon EKS adalah `amazon-sagemaker-hyperpod-taskgovernance`.

Izin IAM yang diperlukan

Add-on ini tidak memerlukan izin apa pun.

Informasi tambahan

Untuk mempelajari lebih lanjut tentang add-on, lihat tata kelola [SageMaker HyperPod tugas](#)

Pengaya SageMaker HyperPod Observabilitas Amazon

Amazon SageMaker HyperPod Observability Add-on menyediakan kemampuan pemantauan dan observabilitas yang komprehensif untuk cluster. HyperPod Add-on ini secara otomatis menyebarkan dan mengelola komponen pemantauan penting termasuk eksportir node, eksportir DCGM, dan eksportir EFA. kube-state-metrics Ini mengumpulkan dan meneruskan metrik ke instans Prometheus (AMP) Amazon Managed Prometheus (AMP) yang ditunjuk pelanggan dan mengekspos titik akhir OTLP untuk metrik khusus dan konsumsi peristiwa dari pekerjaan pelatihan pelanggan.

Add-on ini terintegrasi dengan HyperPod ekosistem yang lebih luas dengan mengikis metrik dari berbagai komponen termasuk add-on Tata Kelola HyperPod Tugas, Operator Pelatihan, Kubeflow, HyperPod dan KEDA. Semua metrik yang dikumpulkan dipusatkan di Prometheus yang Dikelola Amazon, memungkinkan pelanggan mencapai tampilan observabilitas terpadu melalui dasbor Grafana Terkelola Amazon. Ini memberikan end-to-end visibilitas ke dalam kesehatan kluster, pemanfaatan sumber daya, dan kinerja kerja pelatihan di seluruh HyperPod lingkungan.

Nama add-on Amazon EKS adalah `amazon-sagemaker-hyperpod-observability`.

Izin IAM yang diperlukan

Add-on ini menggunakan peran IAM untuk kemampuan akun layanan Amazon EKS. Untuk informasi selengkapnya, lihat [the section called “Kredensial dengan IRSA”](#). Diperlukan kebijakan terkelola berikut:

- `AmazonPrometheusRemoteWriteAccess`- untuk metrik penulisan jarak jauh dari cluster ke AMP
- `CloudWatchAgentServerPolicy`- untuk menulis log jarak jauh dari cluster ke CloudWatch

Informasi tambahan

Untuk mempelajari lebih lanjut tentang add-on dan kemampuannya, lihat [SageMaker HyperPod Observabilitas](#).

AWS Agen Monitor Aliran Jaringan

Amazon CloudWatch Network Flow Monitor Agent adalah aplikasi Kubernetes yang mengumpulkan statistik koneksi TCP dari semua node dalam sebuah cluster dan menerbitkan laporan aliran jaringan ke Amazon Network Flow Monitor Ingestion. CloudWatch APIs

Nama add-on Amazon EKS adalah `aws-network-flow-monitoring-agent`.

Izin IAM yang diperlukan

Add-on ini memang memerlukan izin IAM.

Anda harus melampirkan kebijakan `CloudWatchNetworkFlowMonitorAgentPublishPolicy` terkelola ke add-on.

Untuk informasi selengkapnya tentang penyiapan IAM yang diperlukan, lihat [Kebijakan IAM di repo Agen GitHub Monitor Aliran CloudWatch Jaringan Amazon](#).

Untuk informasi selengkapnya tentang kebijakan terkelola, lihat [CloudWatchNetworkFlowMonitorAgentPublishPolicy](#) di Panduan CloudWatch Pengguna Amazon.

Informasi tambahan

Untuk mempelajari lebih lanjut tentang add-on, lihat [GitHub repo Agen Monitor Aliran CloudWatch Jaringan Amazon](#).

Agen pemantauan simpul

Add-on agen pemantauan node Amazon EKS dapat mendeteksi masalah kesehatan node tambahan. Sinyal kesehatan ekstra ini juga dapat dimanfaatkan oleh fitur perbaikan otomatis node opsional untuk secara otomatis mengganti node sesuai kebutuhan.

Note

Anda tidak perlu menginstal add-on ini di kluster Amazon EKS Auto Mode. Untuk informasi selengkapnya, lihat [the section called “Pertimbangan untuk Mode Otomatis Amazon EKS”](#).

Nama add-on Amazon EKS adalah `eks-node-monitoring-agent`.

Izin IAM yang diperlukan

Add-on ini tidak memerlukan izin tambahan.

Informasi tambahan

Untuk informasi selengkapnya, lihat [the section called “Kesehatan simpul”](#).

AWS Distro untuk OpenTelemetry

Add-on AWS Distro untuk OpenTelemetry Amazon EKS adalah distribusi proyek yang aman, siap produksi, dan AWS didukung. OpenTelemetry Untuk informasi lebih lanjut, lihat [AWS Distro untuk informasi lebih OpenTelemetry lanjut GitHub](#).

Nama add-on Amazon EKS adalah `adot`.

Izin IAM yang diperlukan

Add-on ini hanya memerlukan izin IAM jika Anda menggunakan salah satu sumber daya kustom yang telah dikonfigurasi sebelumnya yang dapat dipilih melalui konfigurasi lanjutan.

Informasi tambahan

Untuk informasi selengkapnya, lihat [Memulai dengan AWS Distro untuk OpenTelemetry menggunakan Eks Add-On](#) di AWS Distro untuk dokumentasi. OpenTelemetry

[ADOT mengharuskan `cert-manager` add-on diterapkan di cluster sebagai prasyarat, jika tidak, add-on ini tidak akan berfungsi jika diterapkan langsung menggunakan `https://registry.terraform.io/modules/terraform-aws-modules/eks/aws/latest` `cluster\_addons` properti.](#) Untuk persyaratan lebih lanjut, lihat [Persyaratan untuk Memulai dengan AWS Distro untuk OpenTelemetry menggunakan Eks Add-On](#) di AWS Distro untuk dokumentasi. OpenTelemetry

GuardDuty Agen Amazon

Add-on Amazon EKS GuardDuty agen Amazon mengumpulkan [peristiwa runtime](#) (akses file, eksekusi proses, koneksi jaringan) dari node kluster EKS Anda untuk dianalisis oleh GuardDuty Runtime Monitoring. GuardDuty sendiri (bukan agen) adalah layanan pemantauan keamanan yang menganalisis dan memproses [sumber data dasar](#) termasuk peristiwa AWS CloudTrail manajemen dan log aliran VPC Amazon, serta [fitur-fitur](#), seperti log audit Kubernetes dan pemantauan runtime.

Nama add-on Amazon EKS adalah `aws-guardduty-agent`.

Izin IAM yang diperlukan

Add-on ini tidak memerlukan izin apa pun.

Informasi tambahan

Untuk informasi selengkapnya, lihat [Runtime Monitoring untuk kluster Amazon EKS di Amazon GuardDuty](#)

- Untuk mendeteksi potensi ancaman keamanan di kluster Amazon EKS Anda, aktifkan pemantauan GuardDuty runtime Amazon dan terapkan agen GuardDuty keamanan ke kluster Amazon EKS Anda.

Agen CloudWatch Observabilitas Amazon

Agen CloudWatch Observabilitas Amazon Amazon EKS menambahkan layanan pemantauan dan observabilitas yang disediakan oleh AWS Add-on ini menginstal CloudWatch Agen dan mengaktifkan Sinyal CloudWatch Aplikasi dan CloudWatch Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS. Untuk informasi selengkapnya, lihat [CloudWatch Agen Amazon](#).

Nama add-on Amazon EKS adalah `amazon-cloudwatch-observability`.

Izin IAM yang diperlukan

Add-on ini menggunakan peran IAM untuk kemampuan akun layanan Amazon EKS. Untuk informasi selengkapnya, lihat [the section called “Kredensial dengan IRSA”](#). Izin di tautan: `iam:/home#/policies/arn:aws:iam::aws:policy/AWSXrayWriteOnlyAccess[AWSXrayWriteOnlyAccess,type="console"]` and link: `iam:/home#/policies/arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy [CloudWatchAgentServerPolicy, type="console"]` kebijakan terkelola diperlukan. AWS Anda dapat membuat peran IAM, melampirkan kebijakan terkelola ke dalamnya, dan membuat anotasi akun layanan Kubernetes yang digunakan oleh add-on dengan perintah berikut. Ganti *my-cluster* dengan nama cluster Anda dan *AmazonEKS_Observability_role* dengan nama untuk peran Anda. Perintah ini mengharuskan Anda menginstal `eksctl` di perangkat Anda. Jika Anda perlu menggunakan alat lain untuk membuat peran, lampirkan kebijakan padanya, dan beri anotasi pada akun layanan Kubernetes, lihat [the section called “Tetapkan peran IAM”](#)

```
eksctl create iamserviceaccount \
  --name cloudwatch-agent \
  --namespace amazon-cloudwatch \
  --cluster my-cluster \
  --role-name AmazonEKS_Observability_Role \
  --role-only \
  --attach-policy-arn arn:aws:iam::aws:policy/AWSXrayWriteOnlyAccess \
  --attach-policy-arn arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy \
  --approve
```

Informasi tambahan

Untuk informasi selengkapnya, lihat [Menginstal CloudWatch agen](#).

AWS Konektor CA Pribadi untuk Kubernetes

Konektor CA AWS Pribadi untuk Kubernetes adalah add-on untuk cert-manager yang memungkinkan pengguna memperoleh Sertifikat dari AWS Private Certificate Authority (Private CA).AWS

- Nama add-on Amazon EKS adalah `aws-privateca-connector-for-kubernetes`.
- Namespace add-on adalah `aws-privateca-issuer`

Add-on ini membutuhkan `cert-manager`. `cert-manager` tersedia di Amazon EKS sebagai add-on komunitas. Untuk informasi selengkapnya tentang add-on ini, lihat [the section called “Manajer sertifikat”](#). Untuk informasi selengkapnya tentang menginstal add-on, lihat [the section called “Buat add-on”](#).

Izin IAM yang diperlukan

Add-on ini memerlukan izin IAM.

Gunakan Identitas Pod EKS untuk melampirkan Kebijakan

`AWSPrivateCAConnectorForKubernetesPolicy` IAM ke Akun Layanan `aws-privateca-issuer` Kubernetes. Untuk informasi selengkapnya, lihat [the section called “Gunakan Identitas Pod”](#).

Untuk informasi tentang izin yang diperlukan, lihat [AWSPrivateCAConnectorForKubernetesPolicy](#) di Referensi Kebijakan AWS Terkelola.

Informasi tambahan

Untuk informasi selengkapnya, lihat [AWS Private CA Issuer for GitHub Kubernetes](#) repository.

Untuk informasi selengkapnya tentang mengonfigurasi add-on, lihat [values.yaml](#) di repo. `aws-privateca-issuer` GitHub Konfirmasikan versi `values.yaml` cocok dengan versi add-on yang diinstal pada cluster Anda.

Add-on ini mentolerir `CriticalAddonsOnly` taint yang digunakan oleh EKS Auto system NodePool Mode. Untuk informasi selengkapnya, lihat [the section called “Jalankan add-on penting”](#).

Agen Identitas EKS Pod

Add-on Amazon EKS Pod Identity Agent Amazon EKS menyediakan kemampuan untuk mengelola kredensial untuk aplikasi Anda, mirip dengan cara profil EC2 instans memberikan kredensial ke instans. EC2

Note

Anda tidak perlu menginstal add-on ini di kluster Amazon EKS Auto Mode. Amazon EKS Auto Mode terintegrasi dengan EKS Pod Identity. Untuk informasi selengkapnya, lihat [the section called “Pertimbangan untuk Mode Otomatis Amazon EKS”](#).

Nama add-on Amazon EKS adalah `eks-pod-identity-agent`.

Izin IAM yang diperlukan

Add-on Pod Identity Agent sendiri tidak memerlukan peran IAM. Ini menggunakan izin dari peran [IAM node Amazon EKS untuk berfungsi, tetapi tidak memerlukan peran](#) IAM khusus untuk add-on.

Perbarui informasi

Anda hanya dapat memperbarui satu versi minor dalam satu waktu. Misalnya, jika versi Anda saat ini `1.28.x-eksbuild.y` dan Anda ingin memperbaruinya `1.30.x-eksbuild.y`, maka Anda harus memperbarui versi Anda saat ini `1.29.x-eksbuild.y` dan kemudian memperbaruinya lagi ke `1.30.x-eksbuild.y`. Untuk informasi selengkapnya tentang memperbarui add-on, lihat [the section called “Perbarui add-on”](#).

Pengaya komunitas

Anda dapat menggunakan AWS APIs untuk menginstal add-on komunitas, seperti Kubernetes Metrics Server. Anda dapat memilih untuk menginstal add-on komunitas sebagai Add-on Amazon EKS untuk mengurangi kompleksitas pemeliharaan perangkat lunak pada beberapa cluster.

Misalnya, Anda dapat menggunakan AWS API, CLI, atau Management Console untuk menginstal add-on komunitas. Anda dapat menginstal add-on komunitas selama pembuatan kluster.

Anda mengelola add-on komunitas seperti Add-on Amazon EKS yang ada. Add-on komunitas berbeda dari add-on yang ada karena mereka memiliki cakupan dukungan yang unik.

 Note

Menggunakan add-on komunitas adalah kebijakan Anda. Sebagai bagian dari [model tanggung jawab bersama](#) antara Anda dan AWS, Anda diharapkan untuk memahami apa yang Anda instal ke cluster Anda untuk plugin pihak ketiga ini. Anda juga bertanggung jawab atas add-on komunitas yang memenuhi kebutuhan keamanan klaster Anda. Untuk informasi selengkapnya, lihat [the section called “Support untuk perangkat lunak yang digunakan untuk EKS”](#).

Add-on komunitas dikemas oleh AWS. AWS hanya memvalidasi add-on komunitas untuk kompatibilitas versi. Misalnya, jika Anda menginstal add-on komunitas di sebuah klaster, AWS memeriksa apakah itu kompatibel dengan versi Kubernetes dari klaster Anda.

Yang penting, AWS tidak memberikan dukungan penuh untuk add-on komunitas. AWS hanya mendukung operasi siklus hidup yang dilakukan menggunakan AWS APIs, seperti menginstal add-on atau menghapus add-on.

Jika Anda memerlukan dukungan untuk add-on komunitas, manfaatkan sumber daya proyek yang ada. Misalnya, Anda dapat membuat GitHub masalah pada repo untuk proyek tersebut.

Tentukan jenis add-on

Anda dapat menggunakan AWS CLI untuk menentukan jenis Add-on Amazon EKS.

Gunakan perintah CLI berikut untuk mengambil informasi tentang add-on. Anda dapat mengganti `metrics-server` dengan nama add-on apa pun.

```
aws eks describe-addon-versions --addon-name metrics-server
```

Tinjau output CLI untuk bidang tersebut. `owner`

```
{
  "addons": [
    {
      "addonName": "metrics-server",
      "type": "observability",
      "owner": "community",
      "addonVersions": [
```

Jika nilainya `owner` adalah `community`, maka add-on adalah add-on komunitas. AWS hanya menyediakan dukungan untuk menginstal, memperbarui, dan menghapus add-on. Jika Anda memiliki pertanyaan tentang fungsionalitas dan pengoperasian add-on itu sendiri, gunakan sumber daya komunitas seperti GitHub masalah.

Instal atau perbarui add-on komunitas

Anda menginstal atau memperbarui add-on komunitas dengan cara yang sama seperti Add-on Amazon EKS lainnya.

- [the section called “Buat add-on”](#)
- [the section called “Perbarui add-on”](#)
- [the section called “Hapus add-on”](#)

Add-on komunitas yang tersedia

Add-on komunitas berikut tersedia dari Amazon EKS.

- [the section called “Server Metrik Kubernetes”](#)
- [the section called “kube-state-metrics”](#)
- [the section called “Prometheus Node Exportir”](#)
- [the section called “Manajer sertifikat”](#)
- [the section called “DNS Eksternal”](#)
- [the section called “Fluent Bit”](#)

Server Metrik Kubernetes

Kubernetes Metrics Server adalah sumber metrik sumber daya kontainer yang dapat diskalakan dan efisien untuk pipeline penskalaan otomatis bawaan Kubernetes. Ini mengumpulkan metrik sumber daya dari Kubelet dan mengeksposnya di apiserver Kubernetes melalui Metrics API untuk digunakan oleh Horizontal Pod Autoscaler dan Vertical Pod Autoscaler.

Properti	Nilai
Nama add-on	<code>metrics-server</code>

Properti	Nilai
Namespace	kube-system
Dokumentasi	GitHub Readme
Nama akun layanan	Tidak ada
Kebijakan IAM terkelola	Tidak ada
Izin IAM kustom	Tidak ada

kube-state-metrics

Add-on agent untuk menghasilkan dan mengekspos metrik tingkat cluster.

Status objek Kubernetes di Kubernetes API dapat diekspos sebagai metrik. Agen add-on yang disebut kube-state-metrics dapat terhubung ke server API Kubernetes dan mengekspos titik akhir HTTP dengan metrik yang dihasilkan dari status objek individual dalam cluster. Ini memaparkan berbagai informasi tentang keadaan objek seperti label dan anotasi, waktu startup dan penghentian, status atau fase objek saat ini berada.

Properti	Nilai
Nama add-on	kube-state-metrics
Namespace	kube-state-metrics
Dokumentasi	Metrik untuk Status Objek Kubernetes di Dokumen Kubernetes
Nama akun layanan	Tidak ada
Kebijakan IAM terkelola	Tidak ada
Izin IAM kustom	Tidak ada

Prometheus Node Exportir

Eksportir Prometheus untuk metrik perangkat keras dan OS yang diekspos oleh kernel *NIX, ditulis dalam Go dengan kolektor metrik yang dapat dicolokkan. Prometheus Node Exporter memperlihatkan berbagai macam metrik terkait perangkat keras dan kernel.

Properti	Nilai
Nama add-on	prometheus-node-exporter
Namespace	prometheus-node-exporter
Dokumentasi	Memantau metrik host Linux dengan Node Exporter di Prometheus Docs
Nama akun layanan	Tidak ada
Kebijakan IAM terkelola	Tidak ada
Izin IAM kustom	Tidak ada

Manajer sertifikat

Cert Manager dapat digunakan untuk mengelola pembuatan dan pembaruan sertifikat.

Properti	Nilai
Nama add-on	cert-manager
Namespace	cert-manager
Dokumentasi	Dokumen Manajer Sertifikat
Nama akun layanan	Tidak ada
Kebijakan IAM terkelola	Tidak ada
Izin IAM kustom	Tidak ada

DNS Eksternal

Add-on EKS DNS Eksternal dapat digunakan untuk mengelola catatan DNS Route53 melalui sumber daya Kubernetes.

Izin DNS eksternal dapat dikurangi menjadi `route53:ChangeResourceRecordSets`, `route53:ListHostedZones`, dan `route53:ListResourceRecordSets` pada zona host yang ingin Anda kelola.

Properti	Nilai
Nama add-on	external-dns
Namespace	external-dns
Dokumentasi	GitHub Readme
Nama akun layanan	external-dns
Kebijakan IAM terkelola	arn:aws:iam::aws:policy/AmazonRoute53FullAccess
Izin IAM kustom	Tidak ada

Fluent Bit

Fluent Bit adalah prosesor log dan forwarder yang ringan dan berkinerja tinggi. Ini memungkinkan Anda untuk mengumpulkan data/logs dari berbagai sumber, menyatukannya, dan mengirimkannya ke beberapa tujuan termasuk Amazon CloudWatch Log, Amazon S3, dan Amazon Kinesis Data Firehose. Fluent Bit dirancang dengan mempertimbangkan kinerja dan efisiensi sumber daya, sehingga ideal untuk lingkungan Kubernetes.

Add-on ini tidak memerlukan izin IAM dalam konfigurasi default. Namun, Anda mungkin perlu memberikan izin IAM add-on ini jika Anda mengonfigurasi lokasi keluaran AWS. Untuk informasi selengkapnya, lihat [the section called “Gunakan Identitas Pod”](#).

Properti	Nilai
Nama add-on	fluent-bit

Properti	Nilai
Namespace	fluent-bit
Dokumentasi	Dokumentasi Bit Lancar
Nama akun layanan	fluent-bit
Kebijakan IAM terkelola	Tidak ada
Izin IAM kustom	Tidak ada

Lihat Atribusi

Anda dapat mengunduh atribusi open source dan informasi lisensi untuk add-on komunitas.

1. Tentukan nama dan versi add-on yang ingin Anda unduh atribusi.
2. Perbarui perintah berikut dengan nama dan versi:

```
curl -O https://amazon-eks-docs.s3.amazonaws.com/attributions/<add-on-name>/<add-on-version>/attributions.zip
```

Misalnya:

```
curl -O https://amazon-eks-docs.s3.amazonaws.com/attributions/kube-state-metrics/v2.14.0-eksbuild.1/attributions.zip
```

3. Gunakan perintah untuk mengunduh file.

Gunakan file zip ini untuk melihat informasi tentang atribusi lisensi.

AWS Pengaya Marketplace

Selain daftar add-on Amazon EKS sebelumnya, Anda juga dapat menambahkan berbagai pilihan perangkat lunak operasional Amazon EKS add-on dari vendor perangkat lunak independen. Pilih add-on untuk mempelajari lebih lanjut tentang hal itu dan persyaratan pemasangannya.

Accuknox

Nama add-on adalah `accuknox_kubearmor` dan namespace adalah `kubearmor`. Accuknox menerbitkan add-on.

Untuk informasi tentang add-on, lihat [Memulai dengan KubeArmor](#) dalam KubeArmor dokumentasi.

Nama akun layanan

Akun layanan tidak digunakan dengan add-on ini.

AWS kebijakan IAM terkelola

Kebijakan terkelola tidak digunakan dengan add-on ini.

Izin IAM kustom

Izin khusus tidak digunakan dengan add-on ini.

Akuitas

Nama add-on adalah `akuity_agent` dan namespace adalah `akuity`. Akuity menerbitkan add-on.

Untuk informasi tentang cara add-on, lihat [Menginstal Agen Akuity di Amazon EKS dengan add-on Akuity EKS di](#) dokumentasi Platform Akuity.

Nama akun layanan

Akun layanan tidak digunakan dengan add-on ini.

AWS kebijakan IAM terkelola

Kebijakan terkelola tidak digunakan dengan add-on ini.

Izin IAM kustom

Izin khusus tidak digunakan dengan add-on ini.

Calyptia

Nama add-on adalah `calyptia_fluent-bit` dan namespace adalah `calyptia-fluentbit`. Calyptia menerbitkan add-on.

Untuk informasi tentang add-on, lihat [Memulai Calyptia Core Agent](#) di situs web dokumentasi Calyptia.

Nama akun layanan

Nama akun layanan adalah `calyptia-fluentbit`.

AWS kebijakan IAM terkelola

Add-on ini menggunakan kebijakan `AWSMarketplaceMeteringRegisterUsage` terkelola. Untuk informasi selengkapnya, lihat [AWSMarketplaceMeteringRegisterUsage](#) di Panduan Referensi Kebijakan AWS Terkelola.

Perintah untuk membuat peran IAM yang diperlukan

Perintah berikut mengharuskan Anda memiliki penyedia IAM OpenID Connect (OIDC) untuk cluster Anda. Untuk menentukan apakah Anda memilikinya, atau membuatnya, lihat [the section called “Penyedia IAM OIDC”](#). Ganti *my-cluster* dengan nama cluster Anda dan *my-calyptia-role* dengan nama untuk peran Anda. Perintah ini mengharuskan Anda menginstal [eksctl](#) di perangkat Anda. Jika Anda perlu menggunakan alat lain untuk membuat peran dan membuat anotasi akun layanan Kubernetes, lihat [the section called “Tetapkan peran IAM”](#)

```
eksctl create iamserviceaccount --name service-account-name --namespace calyptia-fluentbit --cluster my-cluster --role-name my-calyptia-role \
    --role-only --attach-policy-arn arn:aws:iam::aws:policy/AWSMarketplaceMeteringRegisterUsage --approve
```

Kolektor Observabilitas Cisco

Nama add-on adalah `cisco_cisco-cloud-observability-collectors` dan namespace adalah `appdynamics`. Cisco mempublikasikan add-on.

Untuk informasi tentang add-on, lihat [Menggunakan Add-On AWS Marketplace Observability Cisco Cloud](#) dalam dokumentasi Cisco AppDynamics

Nama akun layanan

Akun layanan tidak digunakan dengan add-on ini.

AWS kebijakan IAM terkelola

Kebijakan terkelola tidak digunakan dengan add-on ini.

Izin IAM kustom

Izin khusus tidak digunakan dengan add-on ini.

Operator Observabilitas Cisco

Nama add-on adalah `cisco_cisco-cloud-observability-operators` dan namespace adalah `appdynamics` Cisco menerbitkan add-on.

Untuk informasi tentang add-on, lihat [Menggunakan Add-On AWS Marketplace Observability Cisco Cloud](#) dalam dokumentasi Cisco. AppDynamics

Nama akun layanan

Akun layanan tidak digunakan dengan add-on ini.

AWS kebijakan IAM terkelola

Kebijakan terkelola tidak digunakan dengan add-on ini.

Izin IAM kustom

Izin khusus tidak digunakan dengan add-on ini.

CLOUDSOFT

Nama add-on adalah `cloudsoft_cloudsoft-amp` dan namespace adalah `cloudsoft-amp` CLOUDSOFT menerbitkan add-on.

Untuk informasi tentang add-on, lihat [Amazon EKS ADDON di dokumentasi](#) CLOUDSOFT.

Nama akun layanan

Akun layanan tidak digunakan dengan add-on ini.

AWS kebijakan IAM terkelola

Kebijakan terkelola tidak digunakan dengan add-on ini.

Izin IAM kustom

Izin khusus tidak digunakan dengan add-on ini.

Cribl

Nama add-on adalah `cribl_cribledge` dan namespace adalah `cribledge`. Cribl menerbitkan add-on.

Untuk informasi tentang add-on, lihat [Menginstal Add-on Cribl Amazon EKS untuk Edge di dokumentasi](#) Cribl

Nama akun layanan

Akun layanan tidak digunakan dengan add-on ini.

AWS kebijakan IAM terkelola

Kebijakan terkelola tidak digunakan dengan add-on ini.

Izin IAM kustom

Izin khusus tidak digunakan dengan add-on ini.

Dynatrace

Nama add-on adalah `dynatrace_dynatrace-operator` dan namespace adalah `dynatrace`. Dynatrace menerbitkan add-on.

Untuk informasi tentang add-on, lihat [pemantauan Kubernetes](#) di dokumentasi dynatrace.

Nama akun layanan

Akun layanan tidak digunakan dengan add-on ini.

AWS kebijakan IAM terkelola

Kebijakan terkelola tidak digunakan dengan add-on ini.

Izin IAM kustom

Izin khusus tidak digunakan dengan add-on ini.

Datee

Nama add-on adalah `datree_engine-pro` dan namespace adalah `datree`. Datree menerbitkan add-on.

Untuk informasi tentang add-on, lihat [Amazon EKS-Intergration](#) di dokumentasi Datree.

Nama akun layanan

Nama akun layanan adalah `datree-webhook-server-awssmp`.

AWS kebijakan IAM terkelola

Kebijakan yang dikelola adalah `AWSLicenseManagerConsumptionPolicy`. Untuk informasi selengkapnya, lihat [AWSLicenseManagerConsumptionPolicy](#) di Panduan Referensi Kebijakan AWS Terkelola..

Perintah untuk membuat peran IAM yang diperlukan

Perintah berikut mengharuskan Anda memiliki penyedia IAM OpenID Connect (OIDC) untuk cluster Anda. Untuk menentukan apakah Anda memilikinya, atau membuatnya, lihat [the section called “Penyedia IAM OIDC”](#). Ganti *my-cluster* dengan nama cluster Anda dan *my-datree-role* dengan nama untuk peran Anda. Perintah ini mengharuskan Anda menginstal `eksctl` di perangkat Anda. Jika Anda perlu menggunakan alat lain untuk membuat peran dan membuat anotasi akun layanan Kubernetes, lihat [the section called “Tetapkan peran IAM”](#)

```
eksctl create iamserviceaccount --name datree-webhook-server-awssmp --namespace datree
--cluster my-cluster --role-name my-datree-role \
--role-only --attach-policy-arn arn:aws:iam::aws:policy/service-role/
AWSLicenseManagerConsumptionPolicy --approve
```

Izin kustom

Izin khusus tidak digunakan dengan add-on ini.

Datadog

Nama add-on adalah `datadog_operator` dan namespace adalah `datadog-agent` Datadog menerbitkan add-on.

Untuk informasi tentang add-on, lihat [Menginstal Agen Datadog di Amazon EKS dengan Add-on Operator Datadog dalam dokumentasi Datadog](#).

Nama akun layanan

Akun layanan tidak digunakan dengan add-on ini.

AWS kebijakan IAM terkelola

Kebijakan terkelola tidak digunakan dengan add-on ini.

Izin IAM kustom

Izin khusus tidak digunakan dengan add-on ini.

Penutup tanah

Nama add-on adalah `groundcover_agent` dan namespace adalah `groundcover`. `groundcover` menerbitkan `add-on-groundcover`.

Untuk informasi tentang add-on, lihat [Memasang Groundcover Amazon EKS Add-on di dokumentasi groundcover](#).

Nama akun layanan

Akun layanan tidak digunakan dengan add-on ini.

AWS kebijakan IAM terkelola

Kebijakan terkelola tidak digunakan dengan add-on ini.

Izin IAM kustom

Izin khusus tidak digunakan dengan add-on ini.

Laboratorium Grafana

Nama add-on adalah `grafana-labs_kubernetes-monitoring` dan namespace adalah `monitoring`. Grafana Labs menerbitkan add-on.

Untuk informasi tentang add-on, lihat [Mengonfigurasi Pemantauan Kubernetes sebagai Add-on dengan Amazon EKS di dokumentasi Grafana Labs](#).

Nama akun layanan

Akun layanan tidak digunakan dengan add-on ini.

AWS kebijakan IAM terkelola

Kebijakan terkelola tidak digunakan dengan add-on ini.

Izin IAM kustom

Izin khusus tidak digunakan dengan add-on ini.

Guance

- Penerbit — GUANCE
- Nama – guance_datakit
- Ruang nama — datakit
- Nama akun layanan — Akun layanan tidak digunakan dengan add-on ini.
- AWS kebijakan IAM terkelola — Kebijakan terkelola tidak digunakan dengan add-on ini.
- Izin IAM kustom - Izin khusus tidak digunakan dengan add-on ini.
- Petunjuk penyiapan dan penggunaan — Lihat [Menggunakan add-on Amazon EKS](#) di dokumentasi Guance.

Proksi HA

Namanya haproxy-technologies_kubernetes-ingress-ee dan namespace adalah haproxy-controller HA Proxy menerbitkan add-on.

Untuk informasi tentang add-on, lihat [Amazon EKS-Intergration](#) di dokumentasi Datree.

Nama akun layanan

Nama akun layanan adalah `customer` defined.

AWS kebijakan IAM terkelola

Kebijakan yang dikelola adalah `AWSLicenseManagerConsumptionPolicy`. Untuk informasi selengkapnya, lihat [AWSLicenseManagerConsumptionPolicy](#) di Panduan Referensi Kebijakan AWS Terkelola..

Perintah untuk membuat peran IAM yang diperlukan

Perintah berikut mengharuskan Anda memiliki penyedia IAM OpenID Connect (OIDC) untuk cluster Anda. Untuk menentukan apakah Anda memilikinya, atau membuatnya, lihat [the section called “Penyedia IAM OIDC”](#). Ganti *my-cluster* dengan nama cluster Anda dan *my-haproxy-role* dengan nama untuk peran Anda. Perintah ini mengharuskan Anda menginstal [eksctl](#) di perangkat Anda. Jika Anda perlu menggunakan alat lain untuk membuat peran dan membuat anotasi akun layanan Kubernetes, lihat [the section called “Tetapkan peran IAM”](#)

```
eksctl create iamserviceaccount --name service-account-name --namespace haproxy-
controller --cluster my-cluster --role-name my-haproxy-role \
```

```
--role-only --attach-policy-arn arn:aws:iam::aws:policy/service-role/
AWSLicenseManagerConsumptionPolicy --approve
```

Izin kustom

Izin khusus tidak digunakan dengan add-on ini.

Kpow

Nama add-on adalah `factorhouse_kpow` dan namespace adalah `factorhouse`. Factorhouse menerbitkan add-on.

Untuk informasi tentang add-on, lihat [AWS Marketplace LM di dokumentasi Kpow](#).

Nama akun layanan

Nama akun layanan adalah `kpow`.

AWS kebijakan IAM terkelola

Kebijakan yang dikelola adalah `AWSLicenseManagerConsumptionPolicy`. Untuk informasi selengkapnya, lihat [AWSLicenseManagerConsumptionPolicy](#) di Panduan Referensi Kebijakan AWS Terkelola..

Perintah untuk membuat peran IAM yang diperlukan

Perintah berikut mengharuskan Anda memiliki penyedia IAM OpenID Connect (OIDC) untuk cluster Anda. Untuk menentukan apakah Anda memilikinya, atau membuatnya, lihat [the section called “Penyedia IAM OIDC”](#). Ganti *my-cluster* dengan nama cluster Anda dan *my-kpow-role* dengan nama untuk peran Anda. Perintah ini mengharuskan Anda menginstal [eksctl](#) di perangkat Anda. Jika Anda perlu menggunakan alat lain untuk membuat peran dan membuat anotasi akun layanan Kubernetes, lihat [the section called “Tetapkan peran IAM”](#).

```
eksctl create iamserviceaccount --name kpow --namespace factorhouse --cluster my-
cluster --role-name my-kpow-role \
    --role-only --attach-policy-arn arn:aws:iam::aws:policy/service-role/
AWSLicenseManagerConsumptionPolicy --approve
```

Izin kustom

Izin khusus tidak digunakan dengan add-on ini.

Kubecost

Nama add-on adalah `kubecost_kubecost` dan namespace adalah `kubecost`. Kubecost menerbitkan add-on.

Untuk informasi tentang add-on, lihat [Integrasi Penagihan AWS Cloud](#) di dokumentasi Kubecost.

Anda harus memiliki volume [Store Kubernetes dengan Amazon EBS](#) diinstal pada cluster Anda. Jika tidak, Anda akan menerima kesalahan.

Nama akun layanan

Akun layanan tidak digunakan dengan add-on ini.

AWS kebijakan IAM terkelola

Kebijakan terkelola tidak digunakan dengan add-on ini.

Izin IAM kustom

Izin khusus tidak digunakan dengan add-on ini.

Kasten

Nama add-on adalah `kasten_k10` dan namespace adalah `kasten-io`. Kasten oleh Veeam menerbitkan add-on.

Untuk informasi tentang add-on, lihat [Menginstal K10 saat menggunakan AWS Amazon EKS Add-on di dokumentasi Kasten](#).

Anda harus menginstal driver Amazon EBS CSI di cluster Anda dengan default `StorageClass`.

Nama akun layanan

Nama akun layanan adalah `k10-k10`.

AWS kebijakan IAM terkelola

Kebijakan yang dikelola adalah `AWSLicenseManagerConsumptionPolicy`. Untuk informasi selengkapnya, lihat [AWSLicenseManagerConsumptionPolicy](#) di Panduan Referensi Kebijakan AWS Terkelola..

Perintah untuk membuat peran IAM yang diperlukan

Perintah berikut mengharuskan Anda memiliki penyedia IAM OpenID Connect (OIDC) untuk cluster Anda. Untuk menentukan apakah Anda memilikinya, atau membuatnya, lihat [the section called “Penyedia IAM OIDC”](#). Ganti *my-cluster* dengan nama cluster Anda dan *my-kasten-role* dengan nama untuk peran Anda. Perintah ini mengharuskan Anda menginstal [eksctl](#) di perangkat Anda. Jika Anda perlu menggunakan alat lain untuk membuat peran dan membuat anotasi akun layanan Kubernetes, lihat [the section called “Tetapkan peran IAM”](#)

```
eksctl create iamserviceaccount --name k10-k10 --namespace kasten-io --cluster my-cluster --role-name my-kasten-role \
    --role-only --attach-policy-arn arn:aws:iam::aws:policy/service-role/AWSLicenseManagerConsumptionPolicy --approve
```

Izin kustom

Izin khusus tidak digunakan dengan add-on ini.

Kong

Nama add-on adalah `kong_konnect-ri` dan namespace adalah `kong`. Kong menerbitkan add-on.

Untuk informasi tentang add-on, lihat [Menginstal Add-on Kong Gateway EKS di](#) dokumentasi Kong.

Anda harus memiliki volume [Store Kubernetes dengan Amazon EBS](#) diinstal pada cluster Anda. Jika tidak, Anda akan menerima kesalahan.

Nama akun layanan

Akun layanan tidak digunakan dengan add-on ini.

AWS kebijakan IAM terkelola

Kebijakan terkelola tidak digunakan dengan add-on ini.

Izin IAM kustom

Izin khusus tidak digunakan dengan add-on ini.

LeakSignal

Nama add-on adalah `leaksignal_leakagent` dan namespace adalah `leakagent`. LeakSignal menerbitkan add-on.

Untuk informasi tentang add-on, lihat [https://www.leaksignal.com/docs/LeakAgent/Deployment/ AWS %20eks%20Addon/ \[Instal add-on\]](https://www.leaksignal.com/docs/LeakAgent/Deployment/ AWS %20eks%20Addon/ [Instal add-on]) dalam dokumentasi LeakAgent LeakSignal

Anda harus memiliki volume [Store Kubernetes dengan Amazon EBS](#) diinstal pada cluster Anda. jika tidak, Anda akan menerima kesalahan.

Nama akun layanan

Akun layanan tidak digunakan dengan add-on ini.

AWS kebijakan IAM terkelola

Kebijakan terkelola tidak digunakan dengan add-on ini.

Izin IAM kustom

Izin khusus tidak digunakan dengan add-on ini.

NetApp

Nama add-on adalah netapp_trident-operator dan namespace adalah. trident NetApp menerbitkan add-on.

Untuk informasi tentang add-on, lihat [Mengonfigurasi add-on Trident EKS di dokumentasi](#). NetApp

Nama akun layanan

Akun layanan tidak digunakan dengan add-on ini.

AWS kebijakan IAM terkelola

Kebijakan terkelola tidak digunakan dengan add-on ini.

Izin IAM kustom

Izin khusus tidak digunakan dengan add-on ini.

Relik Baru

Nama add-on adalah new-relic_kubernetes-operator dan namespace adalah. newrelic New Relic menerbitkan add-on.

Untuk informasi tentang add-on, lihat [Menginstal Add-on Relik Baru untuk EKS di dokumentasi](#) New Relic.

Nama akun layanan

Akun layanan tidak digunakan dengan add-on ini.

AWS kebijakan IAM terkelola

Kebijakan terkelola tidak digunakan dengan add-on ini.

Izin IAM kustom

Izin khusus tidak digunakan dengan add-on ini.

Rafay

Nama add-on adalah `rafay-systems_rafay-operator` dan namespace adalah `rafay-system`. Rafay menerbitkan add-on.

Untuk informasi tentang add-on, lihat [Menginstal Add-on Rafay Amazon EKS di dokumentasi](#) Rafay.

Nama akun layanan

Akun layanan tidak digunakan dengan add-on ini.

AWS kebijakan IAM terkelola

Kebijakan terkelola tidak digunakan dengan add-on ini.

Izin IAM kustom

Izin khusus tidak digunakan dengan add-on ini.

Keamanan Rad

- Penerbit - RAD SECURITY
- Nama – `rad-security_rad-security`
- Ruang nama — `ksoc`
- Nama akun layanan — Akun layanan tidak digunakan dengan add-on ini.
- AWS kebijakan IAM terkelola — Kebijakan terkelola tidak digunakan dengan add-on ini.
- Izin IAM kustom - Izin khusus tidak digunakan dengan add-on ini.
- Petunjuk penyiapan dan penggunaan — Lihat [Menginstal Rad Through The AWS Marketplace](#) di dokumentasi Rad Security.

SolarWinds

- Penerbit — SOLARWINDS
- Nama – solarwinds_swo-k8s-collector-addon
- Ruang nama — solarwinds
- Nama akun layanan — Akun layanan tidak digunakan dengan add-on ini.
- AWS kebijakan IAM terkelola — Kebijakan terkelola tidak digunakan dengan add-on ini.
- Izin IAM kustom - Izin khusus tidak digunakan dengan add-on ini.
- Petunjuk penyiapan dan penggunaan — Lihat [Memantau klaster Amazon EKS](#) dalam SolarWinds dokumentasi.

Solo

Nama add-on adalah solo-io_istio-distro dan namespace adalah. istio-system Solo menerbitkan add-on.

Untuk informasi tentang add-on, lihat [Menginstal Istio di dokumentasi Solo.io..](#)

Nama akun layanan

Akun layanan tidak digunakan dengan add-on ini.

AWS kebijakan IAM terkelola

Kebijakan terkelola tidak digunakan dengan add-on ini.

Izin IAM kustom

Izin khusus tidak digunakan dengan add-on ini.

Snyk

- Penerbit — SNYK
- Nama – snyk_runtime-sensor
- Ruang nama — snyk_runtime-sensor
- Nama akun layanan — Akun layanan tidak digunakan dengan add-on ini.
- AWS kebijakan IAM terkelola — Kebijakan terkelola tidak digunakan dengan add-on ini.

- Izin IAM kustom - Izin khusus tidak digunakan dengan add-on ini.
- Petunjuk penyiapan dan penggunaan — Lihat [sensor runtime Snyk](#) di dokumen pengguna Snyk.

Stormforge

Nama add-on adalah `stormforge_optimize-Live` dan namespace adalah `stormforge-system`. Stormforge menerbitkan add-on.

Untuk informasi tentang add-on, lihat [Menginstal StormForge Agen](#) dalam StormForge dokumentasi.

Nama akun layanan

Akun layanan tidak digunakan dengan add-on ini.

AWS kebijakan IAM terkelola

Kebijakan terkelola tidak digunakan dengan add-on ini.

Izin IAM kustom

Izin khusus tidak digunakan dengan add-on ini.

Splunk

Nama add-on adalah `splunk_splunk-otel-collector-chart` dan namespace adalah `splunk-monitoring`. Splunk menerbitkan add-on.

Untuk informasi tentang add-on, lihat [Menginstal add-on Splunk untuk Amazon EKS](#) di dokumentasi Splunk.

Nama akun layanan

Akun layanan tidak digunakan dengan add-on ini.

AWS kebijakan IAM terkelola

Kebijakan terkelola tidak digunakan dengan add-on ini.

Izin IAM kustom

Izin khusus tidak digunakan dengan add-on ini.

Teleportasi

Nama add-on adalah `teleport_teleport` dan namespace adalah `teleport`. Teleport menerbitkan add-on.

Untuk informasi tentang add-on, lihat [Cara Kerja Teleport dalam dokumentasi](#) Teleport.

Nama akun layanan

Akun layanan tidak digunakan dengan add-on ini.

AWS kebijakan IAM terkelola

Kebijakan terkelola tidak digunakan dengan add-on ini.

Izin IAM kustom

Izin khusus tidak digunakan dengan add-on ini.

Tetrat

Nama add-on adalah `tetrade-io_istio-distro` dan namespace adalah `istio-system`. Tetrade io menerbitkan add-on.

Untuk informasi tentang add-on, lihat situs web [Tetrade Istio](#) Distro.

Nama akun layanan

Akun layanan tidak digunakan dengan add-on ini.

AWS kebijakan IAM terkelola

Kebijakan terkelola tidak digunakan dengan add-on ini.

Izin IAM kustom

Izin khusus tidak digunakan dengan add-on ini.

Crossplane Universal Naik

Nama add-on adalah `upbound_universal-crossplane` dan namespace adalah `upbound-system`. Upbound menerbitkan add-on.

Untuk informasi tentang add-on, lihat [Upbound Universal Crossplane \(UXP\)](#) di dokumentasi Upbound.

Nama akun layanan

Akun layanan tidak digunakan dengan add-on ini.

AWS kebijakan IAM terkelola

Kebijakan terkelola tidak digunakan dengan add-on ini.

Izin IAM kustom

Izin khusus tidak digunakan dengan add-on ini.

Menengah Angin

Nama add-on adalah `upwind` dan namespace adalah `upwind`. Upwind menerbitkan add-on.

Untuk informasi tentang add-on, lihat dokumentasi [Upwind](#).

Nama akun layanan

Akun layanan tidak digunakan dengan add-on ini.

AWS kebijakan IAM terkelola

Kebijakan terkelola tidak digunakan dengan add-on ini.

Izin IAM kustom

Izin khusus tidak digunakan dengan add-on ini.

Buat add-on Amazon EKS

Pengaya Amazon EKS adalah perangkat lunak tambahan untuk kluster Amazon EKS. Semua pengaya Amazon EKS:

- Sertakan patch keamanan terbaru dan perbaikan bug.
- Divalidasi oleh AWS untuk bekerja dengan Amazon EKS.
- Kurangi jumlah pekerjaan yang diperlukan untuk mengelola perangkat lunak add-on.

Anda dapat membuat add-on Amazon EKS menggunakan `eksctl`, AWS Management Console, atau AWS CLI. Jika add-on memerlukan peran IAM, lihat detail untuk add-on khusus di add-on [Amazon EKS](#) untuk detail tentang membuat peran.

Prasyarat

Selesaikan hal-hal berikut sebelum Anda membuat add-on:

- Cluster harus ada sebelum Anda membuat add-on untuk itu. Untuk informasi selengkapnya, lihat [the section called “Membuat klaster”](#).
- Periksa apakah add-on Anda memerlukan peran IAM. Untuk informasi selengkapnya, lihat [the section called “Verifikasi kompatibilitas”](#).
- Verifikasi bahwa versi add-on Amazon EKS kompatibel dengan cluster Anda. Untuk informasi selengkapnya, lihat [the section called “Verifikasi kompatibilitas”](#).
- Verifikasi bahwa versi 0.190.0 atau yang lebih baru dari alat baris `eksctl` perintah yang diinstal pada komputer Anda atau AWS CloudShell. Untuk informasi selengkapnya, lihat [Pemasangan](#) di situs web `eksctl`.

Prosedur

Anda dapat membuat add-on Amazon EKS menggunakan `eksctl`, AWS Management Console, atau AWS CLI. Jika add-on memerlukan peran IAM, lihat detail untuk add-on spesifik di add-on [Amazon EKS](#) yang Tersedia AWS untuk detail tentang membuat peran.

Buat add-on (`eksctl`)

1. Lihat nama add-on yang tersedia untuk versi cluster. Ganti **1.33** dengan versi cluster Anda.

```
eksctl utils describe-addon-versions --kubernetes-version 1.33 | grep AddonName
```

Contoh output adalah sebagai berikut.

```
"AddonName": "aws-ebs-csi-driver",
    "AddonName": "coredns",
    "AddonName": "kube-proxy",
    "AddonName": "vpc-cni",
    "AddonName": "adot",
    "AddonName": "dynatrace_dynatrace-operator",
    "AddonName": "upbound_universal-crossplane",
    "AddonName": "teleport_teleport",
    "AddonName": "factorhouse_kpow",
    [...]
```


2. Lihat versi yang tersedia untuk add-on yang ingin Anda buat. Ganti **1.33** dengan versi cluster Anda. Ganti **name-of-addon** dengan nama add-on yang ingin Anda lihat versinya. Nama harus salah satu nama yang dikembalikan pada langkah sebelumnya.

```
eksctl utils describe-addon-versions --kubernetes-version 1.33 --name name-of-addon |  
grep AddonVersion
```

Output berikut adalah contoh dari apa yang dikembalikan untuk add-on bernama `avpc-cni`. Anda dapat melihat bahwa add-on memiliki beberapa versi yang tersedia.

```
"AddonVersions": [  
  "AddonVersion": "v1.12.0-eksbuild.1",  
  "AddonVersion": "v1.11.4-eksbuild.1",  
  "AddonVersion": "v1.10.4-eksbuild.1",  
  "AddonVersion": "v1.9.3-eksbuild.1",
```

- a. Tentukan apakah add-on yang ingin Anda buat adalah add-on Amazon EKS atau AWS Marketplace. AWS Marketplace memiliki add-on pihak ketiga yang mengharuskan Anda menyelesaikan langkah-langkah tambahan untuk membuat add-on.

```
eksctl utils describe-addon-versions --kubernetes-version 1.33 --name name-of-  
addon | grep ProductUrl
```

Jika tidak ada output yang dikembalikan, maka add-on adalah Amazon EKS. Jika output dikembalikan, maka add-on adalah add-on AWS Marketplace. Output berikut adalah untuk add-on bernama `teleport-teleport`.

```
"ProductUrl": "https://aws.amazon.com/marketplace/pp?sku=3bda70bb-566f-4976-806c-  
f96faef18b26"
```

Anda dapat mempelajari lebih lanjut tentang add-on di AWS Marketplace dengan URL yang dikembalikan. Jika add-on memerlukan langganan, Anda dapat berlangganan add-on melalui Marketplace AWS. Jika Anda akan membuat add-on dari AWS Marketplace, maka [prinsipal IAM](#) yang Anda gunakan untuk membuat add-on harus memiliki izin untuk membuat peran terkait layanan. [AWSServiceRoleForAWSLicenseManagerRole](#) Untuk informasi selengkapnya tentang menetapkan izin ke entitas IAM, lihat [Menambahkan dan menghapus izin identitas IAM](#) di Panduan Pengguna IAM.

3. Buat add-on Amazon EKS. Salin perintah dan ganti **user-data** sebagai berikut:

- Ganti *my-cluster* dengan nama klaster Anda.
- Ganti *name-of-addon* dengan nama add-on yang ingin Anda buat.
- Jika Anda menginginkan versi add-on yang lebih awal dari versi terbaru, maka ganti *latest* dengan nomor versi yang dikembalikan dalam output dari langkah sebelumnya yang ingin Anda gunakan.
- Jika add-on menggunakan peran akun layanan, ganti *111122223333* dengan ID akun Anda dan ganti *role-name* dengan nama peran. Untuk petunjuk cara membuat peran untuk akun layanan Anda, lihat dokumentasi untuk add-on yang Anda buat. Untuk daftar add-on, lihat [the section called “AWS add-on”](#). Menentukan peran akun layanan mengharuskan Anda memiliki penyedia IAM OpenID Connect (OIDC) untuk klaster Anda. Untuk menentukan apakah Anda memiliki satu untuk cluster Anda, atau untuk membuatnya, lihat [the section called “Penyedia IAM OIDC”](#).

Jika add-on tidak menggunakan peran akun layanan, hapus `--service-account-role-arn arn:aws:iam::111122223333:role/role-name`.

- Perintah contoh ini menerima konfigurasi versi add-on yang dikelola sendiri yang ada, jika ada. Jika Anda tidak ingin menerima konfigurasi add-on yang dikelola sendiri, hapus opsi. *--force* Jika Anda menghapus opsi, dan add-on Amazon EKS perlu menerima konfigurasi add-on yang dikelola sendiri, maka pembuatan add-on Amazon EKS gagal dengan pesan kesalahan untuk membantu Anda menyelesaikan konflik. Sebelum menentukan opsi ini, pastikan add-on Amazon EKS tidak mengelola pengaturan yang perlu Anda kelola, karena pengaturan tersebut ditimpa dengan opsi ini.

```
eksctl create addon --cluster my-cluster --name name-of-addon --version latest \
--service-account-role-arn arn:aws:iam::111122223333:role/role-name --force
```

Anda dapat melihat daftar semua opsi yang tersedia untuk perintah.

```
eksctl create addon --help
```

Untuk informasi selengkapnya tentang opsi yang tersedia, lihat [Addons](#) dalam `eksctl` dokumentasi.

Buat add-on (AWS Konsol)

1. Buka [konsol Amazon EKS](#).

2. Pada panel navigasi sebelah kiri, pilih Klaster.
3. Pilih nama cluster yang ingin Anda buat add-on.
4. Pilih tab Add-ons.
5. Pilih Get more add-ons
6. Pada halaman Pilih add-on, pilih add-on yang ingin Anda tambahkan ke cluster Anda. Anda dapat menambahkan add-on Amazon EKS dan add-on AWS Marketplace sebanyak yang Anda butuhkan.

Untuk add-on AWS Marketplace, [prinsipal IAM](#) yang Anda gunakan untuk membuat add-on harus memiliki izin untuk membaca hak untuk add-on dari AWS LicenseManager. AWS LicenseManager memerlukan peran [AWSServiceRoleForAWSLicenseManagerRole](#) terkait layanan (SLR) yang memungkinkan AWS sumber daya mengelola lisensi atas nama Anda. SLR adalah persyaratan satu kali, per akun, dan Anda tidak perlu membuat SLR terpisah untuk setiap add-on atau setiap cluster. Untuk informasi selengkapnya tentang menetapkan izin ke [prinsipal IAM](#), lihat [Menambahkan dan menghapus izin identitas IAM](#) di Panduan Pengguna IAM.

Jika add-on AWS Marketplace yang ingin Anda instal tidak terdaftar, Anda dapat mengklik penomoran halaman untuk melihat hasil halaman tambahan atau mencari di kotak pencarian. Dalam opsi Pemfilteran, Anda juga dapat memfilter berdasarkan kategori, vendor, atau model harga dan kemudian memilih add-on dari hasil pencarian. Setelah Anda memilih add-on yang ingin Anda instal, pilih Berikutnya.

7. Pada halaman Configure selected add-ons settings, lakukan hal berikut:
 - a. Pilih Lihat opsi berlangganan untuk membuka formulir Opsi berlangganan. Tinjau detail Harga dan bagian Hukum, lalu pilih tombol Berlangganan untuk melanjutkan.
 - b. Untuk Versi, pilih versi yang ingin Anda instal. Kami merekomendasikan versi yang ditandai terbaru, kecuali setiap add-on yang Anda buat merekomendasikan versi yang berbeda. Untuk menentukan apakah add-on memiliki versi yang direkomendasikan, lihat dokumentasi untuk add-on yang Anda buat. Untuk daftar add-on, lihat [the section called “AWS add-on”](#).
 - c. Anda memiliki dua opsi untuk mengonfigurasi peran untuk add-on: EKS Pod Identities IAM role dan IAM role for service account (IRSA). Ikuti langkah yang sesuai di bawah ini untuk opsi pilihan Anda. Jika semua add-on yang Anda pilih memiliki Memerlukan langganan di bawah Status, pilih Berikutnya. Anda tidak dapat [mengonfigurasi add-on tersebut](#) lebih lanjut sampai Anda berlangganan mereka setelah klaster Anda dibuat. Untuk add-on yang tidak memiliki langganan Memerlukan di bawah Status, lakukan hal berikut:

- i. Untuk peran IAM Pod Identity untuk akun layanan, Anda dapat menggunakan peran IAM EKS Pod Identity yang sudah ada atau membuatnya menggunakan tombol Create Recommended Role. Bidang ini hanya akan memberikan opsi dengan kebijakan kepercayaan yang sesuai. Jika tidak ada peran untuk dipilih, maka Anda tidak memiliki peran yang ada dengan kebijakan kepercayaan yang cocok. Untuk mengonfigurasi peran IAM Identitas Pod EKS untuk akun layanan dari add-on yang dipilih, pilih Buat peran yang direkomendasikan. Wizard pembuatan peran terbuka di jendela terpisah. Wizard akan secara otomatis mengisi informasi peran sebagai berikut. Untuk setiap add-on di mana Anda ingin membuat peran EKS Pod Identity IAM, selesaikan langkah-langkah di wizard IAM sebagai berikut.
 - Pada langkah Pilih entitas tepercaya, opsi AWS layanan untuk EKS dan kasus penggunaan untuk EKS - Pod Identity telah dipilih sebelumnya, dan kebijakan kepercayaan yang sesuai akan diisi secara otomatis untuk add-on. Misalnya, peran akan dibuat dengan kebijakan kepercayaan yang sesuai yang berisi Pods.eks.amazonaws.com IAM Principal sebagaimana dirinci dalam [the section called “Manfaat Identitas EKS Pod”](#) Pilih Berikutnya.
 - Pada langkah Tambah izin, kebijakan terkelola yang sesuai untuk kebijakan peran telah dipilih sebelumnya untuk add-on. Misalnya, untuk add-on Amazon VPC CNI, peran akan dibuat dengan kebijakan AmazonEKS_CNI_Policy terkelola seperti yang dijelaskan dalam [the section called “Plugin Amazon VPC CNI untuk Kubernetes”](#) Pilih Berikutnya.
 - Pada langkah Nama, tinjau, dan buat, di Nama peran, nama peran default secara otomatis diisi untuk add-on. Misalnya, untuk add-on Amazon VPC CNI, peran akan dibuat dengan nama Amazon.EKSPod IdentityAmazon VPCCNIRole Dalam Deskripsi, deskripsi default secara otomatis diisi dengan deskripsi yang sesuai untuk add-on. Misalnya, untuk add-on Amazon VPC CNI, peran akan dibuat dengan deskripsi Memungkinkan pod yang berjalan di kluster Amazon EKS untuk mengakses sumber daya. AWS Dalam kebijakan Trust, lihat kebijakan kepercayaan terisi untuk add-on. Pilih Buat peran.

CATATAN: Mempertahankan nama peran default memungkinkan EKS untuk memilih peran untuk add-on di cluster baru atau saat menambahkan add-on ke cluster yang ada. Anda masih dapat mengganti nama ini dan peran akan tersedia untuk add-on di seluruh cluster Anda, tetapi peran tersebut harus dipilih secara manual dari drop-down.

- ii. Untuk add-on yang tidak memiliki langganan Memerlukan di bawah Status dan di mana Anda ingin mengonfigurasi peran menggunakan IRSA, lihat dokumentasi untuk add-on yang Anda buat untuk membuat kebijakan IAM dan melampirkannya ke peran. Untuk daftar add-on, lihat [the section called “ AWS add-on”](#). Memilih peran IAM mengharuskan Anda memiliki

penyedia IAM OpenID Connect (OIDC) untuk klaster Anda. Untuk menentukan apakah Anda memiliki satu untuk cluster Anda, atau untuk membuatnya, lihat [the section called “Penyedia IAM OIDC”](#).

- iii. Pilih Pengaturan konfigurasi opsional.
 - iv. Jika add-on memerlukan konfigurasi, masukkan di kotak Nilai konfigurasi. Untuk menentukan apakah add-on memerlukan informasi konfigurasi, lihat dokumentasi untuk add-on yang Anda buat. Untuk daftar add-on, lihat [the section called “ AWS add-on”](#).
 - v. Pilih salah satu opsi yang tersedia untuk metode Resolusi Konflik. Jika Anda memilih Override untuk metode Resolusi Konflik, satu atau beberapa pengaturan untuk add-on yang ada dapat ditimpa dengan pengaturan add-on Amazon EKS. Jika Anda tidak mengaktifkan opsi ini dan ada konflik dengan pengaturan yang ada, operasi gagal. Anda dapat menggunakan pesan kesalahan yang dihasilkan untuk memecahkan masalah konflik. Sebelum memilih opsi ini, pastikan add-on Amazon EKS tidak mengelola pengaturan yang perlu Anda kelola sendiri.
 - vi. Pilih Berikutnya.
8. Pada halaman Review dan add, pilih Create. Setelah penginstalan add-on selesai, Anda melihat add-on yang diinstal.
 9. Jika salah satu add-on yang Anda instal memerlukan langganan, selesaikan langkah-langkah berikut:
 - a. Pilih tombol Berlangganan di sudut kanan bawah untuk add-on. Anda dibawa ke halaman untuk add-on di AWS Marketplace. Baca informasi tentang add-on seperti Ikhtisar Produk dan Informasi Harga.
 - b. Pilih tombol Lanjutkan Berlangganan di kanan atas halaman add-on.
 - c. Baca Syarat dan Ketentuan. Jika Anda menyetujuinya, pilih Terima Ketentuan. Mungkin perlu beberapa menit untuk memproses langganan. Saat langganan sedang diproses, tombol Kembali ke Amazon EKS Console berwarna abu-abu.
 - d. Setelah langganan selesai diproses, tombol Kembali ke Amazon EKS Console tidak lagi berwarna abu-abu. Pilih tombol untuk kembali ke tab Add-on konsol Amazon EKS untuk klaster Anda.
 - e. Untuk add-on yang Anda berlangganan, pilih Hapus dan instal ulang lalu pilih Instal ulang add-on. Pemasangan add-on dapat memakan waktu beberapa menit. Saat Instalasi selesai, Anda dapat mengonfigurasi add-on.

Buat add-on (AWS CLI)

1. Anda memerlukan versi 2.12.3 atau yang lebih baru atau versi 1.27.160 atau yang lebih baru dari AWS Command Line Interface (AWS CLI) diinstal dan dikonfigurasi pada perangkat Anda atau AWS CloudShell. Untuk memeriksa versi Anda saat ini, gunakan `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Package manager seperti `yum` atau `apt-get`, atau Homebrew untuk macOS seringkali merupakan beberapa versi di belakang versi terbaru CLI AWS. Untuk menginstal versi terbaru, lihat [Menginstal](#) dan [Konfigurasi cepat dengan aws configure](#) di Panduan Pengguna Antarmuka Baris AWS Perintah. Versi AWS CLI yang diinstal AWS CloudShell mungkin juga beberapa versi di belakang versi terbaru. Untuk memperbaruinya, lihat [Menginstal AWS CLI ke direktori home Anda](#) di AWS CloudShell Panduan Pengguna.
2. Tentukan add-on mana yang tersedia. Anda dapat melihat semua add-on yang tersedia, jenisnya, dan penerbitnya. Anda juga dapat melihat URL untuk add-on yang tersedia melalui AWS Marketplace. Ganti [1.33](#) dengan versi cluster Anda.

```
aws eks describe-addon-versions --kubernetes-version 1.33 \
    --query 'addons[].{MarketplaceProductUrl: marketplaceInformation.productUrl,
    Name: addonName, Owner: owner Publisher: publisher, Type: type}' --output table
```

Contoh output adalah sebagai berikut.

```
-----
|
| DescribeAddonVersions
|
+-----+
+-----+-----+-----+
+-----+
|                               MarketplaceProductUrl                               | Name
|                               | Owner | Publisher | Type |                               |
+-----+-----+-----+-----+
+-----+
| None | aws | eks | storage | aws-ebs-csi-driver
| None | aws | eks | networking | coredns
| None | aws | eks | networking | kube-proxy
+-----+-----+-----+-----+
```

```

| None | aws | eks | networking | vpc-cni
| None | aws | eks | observability | adot
| https://aws.amazon.com/marketplace/pp/prodview-brb73nceicv7u |
dynatrace_dynatrace-operator | aws-marketplace | dynatrace | monitoring
|
| https://aws.amazon.com/marketplace/pp/prodview-uhc2iwi5xysoc | upbound_universal-
crossplane | aws-marketplace | upbound | infra-management |
| https://aws.amazon.com/marketplace/pp/prodview-hd2ydsrgqy4li | teleport_teleport
| aws-marketplace | teleport | policy-management |
| https://aws.amazon.com/marketplace/pp/prodview-vgghgqdsplhvc | factorhouse_kpow
| aws-marketplace | factorhouse | monitoring |
| [...] | [...] | [...] | [...] | [...]
+-----+
+-----+-----+-----+
+-----+

```

Output Anda mungkin berbeda. Dalam contoh keluaran ini, ada tiga add-on berbeda yang tersedia dari jenis `networking` dan lima add-on dengan penerbit tipe. `eks` Add-on dengan `aws-marketplace` di Owner kolom mungkin memerlukan langganan sebelum Anda dapat menginstalnya. Anda dapat mengunjungi URL untuk mempelajari lebih lanjut tentang add-on dan berlangganan.

- Anda dapat melihat versi mana yang tersedia untuk setiap add-on. Ganti `1.33` dengan versi cluster Anda dan ganti `vpc-cni` dengan nama add-on yang dikembalikan pada langkah sebelumnya.

```
aws eks describe-addon-versions --kubernetes-version 1.33 --addon-name vpc-cni \
  --query 'addons[].addonVersions[].{Version: addonVersion, Defaultversion:
compatibilities[0].defaultVersion}' --output table
```

Contoh output adalah sebagai berikut.

```

-----
| DescribeAddonVersions |
+-----+-----+
| Defaultversion | Version |
+-----+-----+
| False | v1.12.0-eksbuild.1 |
| True | v1.11.4-eksbuild.1 |

```

	False		v1.10.4-eksbuild.1	
	False		v1.9.3-eksbuild.1	
+-----+-----+				

Versi dengan True di Defaultversion kolom adalah versi yang add-on dibuat dengan, secara default.

4. (Opsional) Temukan opsi konfigurasi untuk add-on yang Anda pilih dengan menjalankan perintah berikut:

```
aws eks describe-addon-configuration --addon-name vpc-cni --addon-version v1.12.0-eksbuild.1
```

```
{
  "addonName": "vpc-cni",
  "addonVersion": "v1.12.0-eksbuild.1",
  "configurationSchema": "{ \"$ref\": \"#/definitions/VpcCni\", \"$schema\": \"http://json-schema.org/draft-06/schema#\", \"definitions\": { \"Cri\": { \"additionalProperties\": false, \"properties\": { \"hostPath\": { \"$ref\": \"#/definitions/HostPath\" } }, \"title\": \"Cri\", \"type\": \"object\" }, \"Env\": { \"additionalProperties\": false, \"properties\": { \"ADDITIONAL_ENI_TAGS\": { \"type\": \"string\" }, \"AWS_VPC_CNI_NODE_PORT_SUPPORT\": { \"format\": \"boolean\", \"type\": \"string\" }, \"AWS_VPC_ENI_MTU\": { \"format\": \"integer\", \"type\": \"string\" }, \"AWS_VPC_K8S_CNI_CONFIGURE_RPFILTER\": { \"format\": \"boolean\", \"type\": \"string\" }, \"AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG\": { \"format\": \"boolean\", \"type\": \"string\" }, \"AWS_VPC_K8S_CNI_EXTERNALSNAT\": { \"format\": \"boolean\", \"type\": \"string\" }, \"AWS_VPC_K8S_CNI_LOGLEVEL\": { \"type\": \"string\" }, \"AWS_VPC_K8S_CNI_LOG_FILE\": { \"type\": \"string\" }, \"AWS_VPC_K8S_CNI_RANDOMIZESNAT\": { \"type\": \"string\" }, \"AWS_VPC_K8S_CNI_VETHPREFIX\": { \"type\": \"string\" }, \"AWS_VPC_K8S_PLUGIN_LOG_FILE\": { \"type\": \"string\" }, \"AWS_VPC_K8S_PLUGIN_LOG_LEVEL\": { \"type\": \"string\" }, \"DISABLE_INTROSPECTION\": { \"format\": \"boolean\", \"type\": \"string\" }, \"DISABLE_METRICS\": { \"format\": \"boolean\", \"type\": \"string\" }, \"DISABLE_NETWORK_RESOURCE_PROVISIONING\": { \"format\": \"boolean\", \"type\": \"string\" }, \"ENABLE_POD_ENI\": { \"format\": \"boolean\", \"type\": \"string\" }, \"ENABLE_PREFIX_DELEGATION\": { \"format\": \"boolean\", \"type\": \"string\" }, \"WARM_ENI_TARGET\": { \"format\": \"integer\", \"type\": \"string\" }, \"WARM_PREFIX_TARGET\": { \"format\": \"integer\", \"type\": \"string\" } }, \"title\": \"Env\", \"type\": \"object\" }, \"HostPath\": { \"additionalProperties\": false, \"properties\": { \"path\": { \"type\": \"string\" } }, \"title\": \"HostPath\", \"type\": \"object\" }, \"Limits\": { \"additionalProperties\": false, \"properties\": { \"cpu\": { \"type\": \"string\" }, \"memory\": { \"type\": \"string\" } }, \"title\": \"Limits\", \"type\": \"object\" }, \"Resources\": { \"additionalProperties\": false, \"properties\": { \"limits\": { \"$ref\": \"#/definitions/Limits\" }, \"requests\": { \"$ref\": \"#/definitions/Limits\" } }, \"title
```



```
\":\\"Resources\\",\\"type\\":\\"object\\"},\\"VpcCni\\":{\\"additionalProperties\\":false,
\\"properties\\":{\\"cri\\":{\\"$ref\\":\\"#/definitions/Cri\\"},\\"env\\":{\\"$ref\\":\\"#/
definitions/Env\\"},\\"resources\\":{\\"$ref\\":\\"#/definitions/Resources\\"}},\\"title\\":
\\"VpcCni\\",\\"type\\":\\"object\\"}}}"
}
```

Outputnya adalah skema JSON standar.

Berikut adalah contoh nilai konfigurasi yang valid, dalam format JSON, yang bekerja dengan skema di atas.

```
{
  "resources": {
    "limits": {
      "cpu": "100m"
    }
  }
}
```

Berikut adalah contoh nilai konfigurasi yang valid, dalam format YAMAL, yang bekerja dengan skema di atas.

```
resources:
  limits:
    cpu: 100m
```

5. Tentukan apakah add-on memerlukan izin IAM. Jika demikian, Anda perlu (1) menentukan apakah Anda ingin menggunakan EKS Pod Identities atau IAM Roles for Service Accounts (IRSA), (2) menentukan ARN dari peran IAM yang akan digunakan dengan add-on, dan (3) menentukan nama akun layanan Kubernetes yang digunakan oleh add-on. Untuk informasi selengkapnya, lihat [the section called “Ambil informasi IAM”](#).

- Amazon EKS menyarankan untuk menggunakan EKS Pod Identities jika add-on mendukungnya. Ini memerlukan [Pod Identity Agent diinstal pada kluster Anda](#). Untuk informasi selengkapnya tentang penggunaan Identitas Pod dengan Add-on, lihat [the section called “Peran IAM”](#)
- Jika add-on atau kluster Anda tidak diatur untuk EKS Pod Identities, gunakan IRSA. [Konfirmasikan IRSA diatur di cluster Anda](#).
- [Tinjau dokumentasi Add-on Amazon EKS untuk menentukan apakah add-on tersebut memerlukan izin IAM dan nama akun layanan Kubernetes terkait](#).

- a. Buat add-on Amazon EKS. Salin perintah yang mengikuti ke perangkat Anda. Buat modifikasi berikut pada perintah sesuai kebutuhan dan kemudian jalankan perintah yang dimodifikasi:
 - Ganti *my-cluster* dengan nama kluster Anda.
 - Ganti *vpc-cni* dengan nama add-on yang dikembalikan dalam output dari langkah sebelumnya yang ingin Anda buat.
 - Ganti *version-number* dengan versi yang dikembalikan dalam output dari langkah sebelumnya yang ingin Anda gunakan.
 - Jika add-on tidak memerlukan izin IAM, hapus. *<service-account-configuration>*
 - Lakukan salah satu tindakan berikut:
 - Jika add-on (1) memerlukan izin IAM, dan (2) kluster Anda menggunakan EKS Pod Identities, ganti *<service-account-configuration>* dengan asosiasi identitas pod berikut. Ganti *<service-account-name>* dengan nama akun layanan yang digunakan oleh add-on. Ganti *<role-arn>* dengan ARN dari peran IAM. Peran tersebut harus memiliki kebijakan kepercayaan yang disyaratkan oleh EKS Pod Identities.

```
--pod-identity-associations 'serviceAccount=<service-account-name>,roleArn=<role-arn>'
```

- Jika add-on (1) memerlukan izin IAM, dan (2) cluster Anda menggunakan IRSA, ganti *<service-account-configuration>* dengan konfigurasi IRSA berikut. Ganti *111122223333* dengan ID akun Anda dan *role-name* dengan nama peran IAM yang ada yang telah Anda buat. Untuk petunjuk tentang cara membuat peran, lihat dokumentasi untuk add-on yang Anda buat. Untuk daftar add-on, lihat [the section called “AWS add-on”](#). Menentukan peran akun layanan mengharuskan Anda memiliki penyedia IAM OpenID Connect (OIDC) untuk kluster Anda. Untuk menentukan apakah Anda memiliki satu untuk cluster Anda, atau untuk membuatnya, lihat [the section called “Penyedia IAM OIDC”](#).

```
--service-account-role-arn arn:aws::iam::111122223333:role/role-name
```

- Contoh perintah ini menerima *--configuration-values* opsi dari versi add-on yang dikelola sendiri yang ada, jika ada. Ganti ini dengan nilai konfigurasi yang diinginkan, seperti string atau input file. Jika Anda tidak ingin memberikan nilai konfigurasi, hapus *--configuration-values* opsi. Jika Anda tidak ingin AWS CLI menerima konfigurasi add-on yang dikelola sendiri, hapus opsi. *--resolve-conflicts OVERWRITE* Jika Anda menghapus opsi, dan add-on Amazon EKS perlu menerima konfigurasi add-on yang dikelola sendiri, maka pembuatan add-on Amazon EKS gagal dengan pesan kesalahan untuk membantu Anda menyelesaikan konflik.

Sebelum menentukan opsi ini, pastikan add-on Amazon EKS tidak mengelola pengaturan yang perlu Anda kelola, karena pengaturan tersebut ditimpa dengan opsi ini.

```
aws eks create-addon --cluster-name my-cluster --addon-name vpc-cni --addon-version
version-number \
    <service-account-configuration> --configuration-values '{"resources":
{"limits":{"cpu":"100m"}}}' --resolve-conflicts OVERWRITE
```

```
aws eks create-addon --cluster-name my-cluster --addon-name vpc-cni --addon-version
version-number \
    <service-account-configuration> --configuration-values 'file://example.yaml' --
resolve-conflicts OVERWRITE
```

Untuk daftar lengkap opsi yang tersedia, lihat [create-addon](#) di Referensi Baris Perintah Amazon EKS. Jika add-on yang Anda buat telah aws-marketplace terdaftar di Owner kolom langkah sebelumnya, maka pembuatan mungkin gagal, dan Anda mungkin menerima pesan kesalahan yang mirip dengan kesalahan berikut.

```
{
  "addon": {
    "addonName": "addon-name",
    "clusterName": "my-cluster",
    "status": "CREATE_FAILED",
    "addonVersion": "version",
    "health": {
      "issues": [
        {
          "code": "AddonSubscriptionNeeded",
          "message": "You are currently not subscribed to this add-on. To
subscribe, visit the AWS Marketplace console, agree to the seller EULA, select the
pricing type if required, then re-install the add-on"
        }
      ]
    }
  }
}
```

Jika Anda menerima kesalahan yang mirip dengan kesalahan pada output sebelumnya, kunjungi URL di output dari langkah sebelumnya untuk berlangganan add-on. Setelah berlangganan, jalankan create-addon perintah lagi.

Perbarui add-on Amazon EKS

Amazon EKS tidak secara otomatis memperbarui add-on ketika versi baru dirilis atau setelah Anda memperbarui klaster Anda ke versi minor Kubernetes yang baru. Untuk memperbarui add-on untuk cluster yang ada, Anda harus memulai pembaruan. Setelah Anda memulai pembaruan, Amazon EKS memperbarui add-on untuk Anda. Sebelum memperbarui add-on, tinjau dokumentasi saat ini untuk add-on. Untuk daftar add-on yang tersedia, lihat [the section called “AWS add-on”](#). Jika add-on memerlukan peran IAM, lihat detail untuk add-on spesifik di add-on [Amazon EKS](#) yang Tersedia AWS untuk detail tentang membuat peran.

Prasyarat

Selesaikan hal-hal berikut sebelum Anda membuat add-on:

- Periksa apakah add-on Anda memerlukan peran IAM. Untuk informasi selengkapnya, lihat [the section called “Add-on Amazon EKS”](#).
- Verifikasi bahwa versi add-on Amazon EKS kompatibel dengan cluster Anda. Untuk informasi selengkapnya, lihat [the section called “Verifikasi kompatibilitas”](#).

Prosedur

Anda dapat memperbarui add-on Amazon EKS menggunakan `eksctl`, AWS Management Console, atau AWS CLI.

Perbarui add-on (eksctl)

1. Tentukan versi add-on dan add-on saat ini yang diinstal pada cluster Anda. Ganti *my-cluster* dengan nama klaster Anda.

```
eksctl get addon --cluster my-cluster
```

Contoh output adalah sebagai berikut.

NAME	VERSION	STATUS	ISSUES	IAMROLE	UPDATE AVAILABLE
coredns	v1.8.7-eksbuild.2	ACTIVE	0		
kube-proxy	v1.23.7-eksbuild.1	ACTIVE	0		v1.23.8-eksbuild.2
vpc-cni	v1.10.4-eksbuild.1	ACTIVE	0		v1.12.0-eksbuild.1, v1.11.4-eksbuild.1, v1.11.3-eksbuild.1, v1.11.2-eksbuild.1, v1.11.0-eksbuild.1

Output Anda mungkin terlihat berbeda, tergantung pada add-on dan versi yang Anda miliki di cluster Anda. Anda dapat melihat bahwa dalam contoh keluaran sebelumnya, dua add-on yang ada di cluster memiliki versi yang lebih baru yang tersedia di UPDATE AVAILABLE kolom.

2. Perbarui add-on.

a. Salin perintah yang mengikuti ke perangkat Anda. Buat modifikasi berikut pada perintah sesuai kebutuhan:

- Ganti *my-cluster* dengan nama klaster Anda.
- Ganti *region-code* dengan AWS Wilayah tempat klaster Anda berada.
- Ganti *vpc-cni* dengan nama add-on yang dikembalikan dalam output dari langkah sebelumnya yang ingin Anda perbarui.
- Jika Anda ingin memperbarui ke versi lebih awal dari versi terbaru yang tersedia, maka ganti *latest* dengan nomor versi yang dikembalikan dalam output dari langkah sebelumnya yang ingin Anda gunakan. Beberapa add-on memiliki versi yang direkomendasikan. Untuk informasi selengkapnya, lihat dokumentasi untuk add-on yang Anda perbarui. Untuk daftar add-on, lihat [the section called “AWS add-on”](#). * Jika add-on menggunakan akun layanan Kubernetes dan peran IAM, ganti *111122223333* dengan ID akun Anda dan *role-name* dengan nama peran IAM yang sudah ada yang telah Anda buat. Untuk petunjuk tentang cara membuat peran, lihat dokumentasi untuk add-on yang Anda buat. Untuk daftar add-on, lihat [the section called “AWS add-on”](#). Menentukan peran akun layanan mengharuskan Anda memiliki penyedia IAM OpenID Connect (OIDC) untuk klaster Anda. Untuk menentukan apakah Anda memiliki satu untuk cluster Anda, atau untuk membuatnya, lihat [the section called “Penyedia IAM OIDC”](#).

Jika add-on tidak menggunakan akun layanan Kubernetes dan peran IAM, hapus baris tersebut. `serviceAccountRoleARN: arn:aws: iam::111122223333:role/role-name`

- *preserve* Opsi mempertahankan nilai yang ada untuk add-on. Jika Anda telah menetapkan nilai kustom untuk pengaturan add-on, dan Anda tidak menggunakan opsi ini, Amazon EKS menimpa nilai Anda dengan nilai defaultnya. Jika Anda menggunakan opsi ini, sebaiknya Anda menguji setiap bidang dan perubahan nilai pada klaster non-produksi sebelum memperbarui add-on pada cluster produksi Anda. Jika Anda mengubah nilai ini menjadi `overwrite`, semua pengaturan diubah ke nilai default Amazon EKS. Jika Anda telah menetapkan nilai kustom untuk setelan apa pun, nilai tersebut mungkin akan ditimpa dengan nilai default Amazon EKS. Jika Anda mengubah nilai ini menjadi `none`, Amazon EKS tidak mengubah

nilai pengaturan apa pun, tetapi pembaruan mungkin gagal. Jika pembaruan gagal, Anda menerima pesan galat untuk membantu menyelesaikan konflik.

```
cat >update-addon.yaml <<EOF
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: my-cluster
  region: region-code

addons:
- name: vpc-cni
  version: latest
  serviceAccountRoleARN: arn:aws:iam::111122223333:role/role-name
  resolveConflicts: preserve
EOF
```

- b. Jalankan perintah yang dimodifikasi untuk membuat `update-addon.yaml` file.
- c. Terapkan file konfigurasi ke cluster Anda.

```
eksctl update addon -f update-addon.yaml
```

Untuk informasi selengkapnya tentang memperbarui add-on, lihat [Memperbarui addons](#) dalam dokumentasi. eksctl

Perbarui add-on (AWS Konsol)

1. Buka [konsol Amazon EKS](#).
2. Pada panel navigasi sebelah kiri, pilih Klaster.
3. Pilih nama cluster yang ingin Anda perbarui add-on.
4. Pilih tab Add-ons.
5. Pilih add-on yang ingin Anda perbarui.
6. Pilih Edit.
7. Pada *name of addon* halaman Konfigurasi, lakukan hal berikut:
 - a. Pilih Versi yang ingin Anda gunakan. Add-on mungkin memiliki versi yang direkomendasikan. Untuk informasi selengkapnya, lihat dokumentasi untuk add-on yang Anda perbarui. Untuk daftar add-on, lihat [the section called “AWS add-on”](#).

- b. Anda memiliki dua opsi untuk mengonfigurasi peran untuk add-on: EKS Pod Identities IAM role dan IAM role for service account (IRSA). Ikuti langkah yang sesuai di bawah ini untuk opsi pilihan Anda. Jika semua add-on yang Anda pilih memiliki Memerlukan langganan di bawah Status, pilih Berikutnya. Untuk add-on yang tidak memiliki langganan Memerlukan di bawah Status, lakukan hal berikut:
- i. Untuk peran IAM Pod Identity untuk akun layanan, Anda dapat menggunakan peran IAM EKS Pod Identity yang sudah ada atau membuatnya menggunakan tombol Create Recommended Role. Bidang ini hanya akan memberikan opsi dengan kebijakan kepercayaan yang sesuai. Jika tidak ada peran untuk dipilih, maka Anda tidak memiliki peran yang ada dengan kebijakan kepercayaan yang cocok. Untuk mengonfigurasi peran IAM Identitas Pod EKS untuk akun layanan dari add-on yang dipilih, pilih Buat peran yang direkomendasikan. Wizard pembuatan peran terbuka di jendela terpisah. Wizard akan secara otomatis mengisi informasi peran sebagai berikut. Untuk setiap add-on di mana Anda ingin membuat peran EKS Pod Identity IAM, selesaikan langkah-langkah di wizard IAM sebagai berikut.
- Pada langkah Pilih entitas tepercaya, opsi AWS layanan untuk EKS dan kasus penggunaan untuk EKS - Pod Identity telah dipilih sebelumnya, dan kebijakan kepercayaan yang sesuai akan diisi secara otomatis untuk add-on. Misalnya, peran akan dibuat dengan kebijakan kepercayaan yang sesuai yang berisi Pods.eks.amazonaws.com IAM Principal sebagaimana dirinci dalam [the section called “Manfaat Identitas EKS Pod”](#) Pilih Berikutnya.
 - Pada langkah Tambah izin, kebijakan terkelola yang sesuai untuk kebijakan peran telah dipilih sebelumnya untuk add-on. Misalnya, untuk add-on Amazon VPC CNI, peran akan dibuat dengan kebijakan AmazonEKS_CNI_Policy terkelola seperti yang dijelaskan dalam [the section called “Plugin Amazon VPC CNI untuk Kubernetes”](#) Pilih Berikutnya.
 - Pada langkah Nama, tinjau, dan buat, di Nama peran, nama peran default secara otomatis diisi untuk add-on. Misalnya, untuk add-on Amazon VPC CNI, peran akan dibuat dengan nama Amazon. EKSPod IdentityAmazon VPCCNIRole Dalam Deskripsi, deskripsi default secara otomatis diisi dengan deskripsi yang sesuai untuk add-on. Misalnya, untuk add-on Amazon VPC CNI, peran akan dibuat dengan deskripsi Memungkinkan pod yang berjalan di kluster Amazon EKS untuk mengakses sumber daya. AWS Dalam kebijakan Trust, lihat kebijakan kepercayaan terisi untuk add-on. Pilih Buat peran.

 Note

Mempertahankan nama peran default memungkinkan EKS untuk memilih peran untuk add-on di cluster baru atau saat menambahkan add-on ke cluster yang ada. Anda masih dapat mengganti nama ini dan peran akan tersedia untuk add-on di seluruh cluster Anda, tetapi peran tersebut harus dipilih secara manual dari drop-down.

- ii. Untuk add-on yang tidak memiliki langganan Memerlukan di bawah Status dan di mana Anda ingin mengonfigurasi peran menggunakan IRSA, lihat dokumentasi untuk add-on yang Anda buat untuk membuat kebijakan IAM dan melampirkannya ke peran. Untuk daftar add-on, lihat [the section called “AWS add-on”](#). Memilih peran IAM mengharuskan Anda memiliki penyedia IAM OpenID Connect (OIDC) untuk klaster Anda. Untuk menentukan apakah Anda memiliki satu untuk cluster Anda, atau untuk membuatnya, lihat [the section called “Penyedia IAM OIDC”](#).
 - c. Perluas pengaturan konfigurasi opsional.
 - d. Dalam nilai Konfigurasi, masukkan informasi konfigurasi khusus add-on apa pun. Untuk informasi selengkapnya, lihat dokumentasi untuk add-on yang Anda perbarui. Untuk daftar add-on, lihat [the section called “AWS add-on”](#)... Untuk metode resolusi Konflik, pilih salah satu opsi. Jika Anda telah menetapkan nilai kustom untuk pengaturan add-on, kami merekomendasikan opsi Preserve. Jika Anda tidak memilih opsi ini, Amazon EKS menimpa nilai Anda dengan nilai defaultnya. Jika Anda menggunakan opsi ini, sebaiknya Anda menguji setiap bidang dan perubahan nilai pada klaster non-produksi sebelum memperbarui add-on pada cluster produksi Anda. Jika Anda mengubah nilai ini menjadi menimpa, semua pengaturan diubah ke nilai default Amazon EKS. Jika Anda telah menetapkan nilai kustom untuk setelan apa pun, nilai tersebut mungkin akan ditimpa dengan nilai default Amazon EKS. Jika Anda mengubah nilai ini menjadi tidak ada, Amazon EKS tidak mengubah nilai pengaturan apa pun, tetapi pembaruan mungkin gagal. Jika pembaruan gagal, Anda menerima pesan galat untuk membantu menyelesaikan konflik.
8. Pilih Simpan perubahan.

Perbarui add-on (AWS CLI)

1. Anda memerlukan versi 2.12.3 atau yang lebih baru atau versi 1.27.160 atau yang lebih baru dari AWS Command Line Interface (AWS CLI) diinstal dan dikonfigurasi pada perangkat Anda

atau. AWS CloudShell Untuk memeriksa versi Anda saat ini, gunakan `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Package manager seperti `yum` atau `apt-get`, atau Homebrew untuk macOS seringkali merupakan beberapa versi di belakang versi terbaru CLI. AWS Untuk menginstal versi terbaru, lihat [Menginstal](#) dan [Konfigurasi cepat dengan aws configure](#) di Panduan Pengguna Antarmuka Baris AWS Perintah. Versi AWS CLI yang diinstal AWS CloudShell mungkin juga beberapa versi di belakang versi terbaru. Untuk memperbaruinya, lihat [Menginstal AWS CLI ke direktori home Anda](#) di AWS CloudShell Panduan Pengguna.

2. Lihat daftar add-on yang diinstal. Ganti *my-cluster* dengan nama kluster Anda.

```
aws eks list-addons --cluster-name my-cluster
```

Contoh output adalah sebagai berikut.

```
{
  "addons": [
    "coredns",
    "kube-proxy",
    "vpc-cni"
  ]
}
```

3. Lihat versi add-on saat ini yang ingin Anda perbarui. Ganti *my-cluster* dengan nama cluster Anda dan *vpc-cni* dengan nama add-on yang ingin Anda perbarui.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
"addon.addonVersion" --output text
```

Contoh output adalah sebagai berikut.

```
v1.10.4-eksbuild.1
```

4. Tentukan versi add-on mana yang tersedia untuk versi kluster Anda. Ganti *1.33* dengan versi cluster Anda dan *vpc-cni* dengan nama add-on yang ingin Anda perbarui.

```
aws eks describe-addon-versions --kubernetes-version 1.33 --addon-name vpc-cni \
--query 'addons[].addonVersions[].{Version: addonVersion, Defaultversion:
compatibilities[0].defaultVersion}' --output table
```

Contoh output adalah sebagai berikut.

```

-----
|           DescribeAddonVersions           |
+-----+-----+
| Defaultversion |           Version           |
+-----+-----+
|   False       | v1.12.0-eksbuild.1         |
|   True        | v1.11.4-eksbuild.1         |
|   False       | v1.10.4-eksbuild.1         |
|   False       | v1.9.3-eksbuild.1          |
+-----+-----+

```

Versi dengan True di Defaultversion kolom adalah versi yang add-on dibuat dengan, secara default.

5. Perbarui add-on Anda. Salin perintah yang mengikuti ke perangkat Anda. Buat modifikasi berikut pada perintah, sesuai kebutuhan, dan kemudian jalankan perintah yang dimodifikasi. Untuk informasi selengkapnya tentang perintah ini, lihat [update-addon di Referensi Baris Perintah Amazon EKS](#).

- Ganti *my-cluster* dengan nama klaster Anda.
- Ganti *vpc-cni* dengan nama add-on yang ingin Anda perbarui yang dikembalikan dalam output dari langkah sebelumnya.
- Ganti *version-number* dengan versi yang dikembalikan dalam output dari langkah sebelumnya yang ingin Anda perbarui. Beberapa add-on memiliki versi yang direkomendasikan. Untuk informasi selengkapnya, lihat dokumentasi untuk add-on yang Anda perbarui. Untuk daftar add-on, lihat [the section called “AWS add-on”](#). * Jika add-on menggunakan akun layanan Kubernetes dan peran IAM, ganti *111122223333* dengan ID akun Anda dan *role-name* dengan nama peran IAM yang sudah ada yang telah Anda buat. Untuk petunjuk tentang cara membuat peran, lihat dokumentasi untuk add-on yang Anda buat. Untuk daftar add-on, lihat [the section called “AWS add-on”](#). Menentukan peran akun layanan mengharuskan Anda memiliki penyedia IAM OpenID Connect (OIDC) untuk klaster Anda. Untuk menentukan apakah Anda memiliki satu untuk cluster Anda, atau untuk membuatnya, lihat [the section called “Penyedia IAM OIDC”](#).

Jika add-on tidak menggunakan akun layanan Kubernetes dan peran IAM, hapus baris tersebut. `serviceAccountRoleARN: arn:aws: iam::111122223333:role/role-name`

- `--resolve-conflicts PRESERVE` Opsi mempertahankan nilai yang ada untuk add-on. Jika Anda telah menetapkan nilai kustom untuk pengaturan add-on, dan Anda tidak menggunakan

opsi ini, Amazon EKS menimpa nilai Anda dengan nilai defaultnya. Jika Anda menggunakan opsi ini, sebaiknya Anda menguji setiap bidang dan perubahan nilai pada kluster non-produksi sebelum memperbarui add-on pada cluster produksi Anda. Jika Anda mengubah nilai ini menjadi `OVERWRITE`, semua pengaturan diubah ke nilai default Amazon EKS. Jika Anda telah menetapkan nilai kustom untuk setelan apa pun, nilai tersebut mungkin akan ditimpa dengan nilai default Amazon EKS. Jika Anda mengubah nilai ini menjadi `NONE`, Amazon EKS tidak mengubah nilai pengaturan apa pun, tetapi pembaruan mungkin gagal. Jika pembaruan gagal, Anda menerima pesan galat untuk membantu menyelesaikan konflik.

- Jika Anda ingin menghapus semua konfigurasi kustom maka lakukan pembaruan menggunakan `--configuration-values '{}'` opsi. Ini menetapkan semua konfigurasi kustom kembali ke nilai default. Jika Anda tidak ingin mengubah konfigurasi kustom Anda, jangan berikan `--configuration-values` tanda. Jika Anda ingin menyesuaikan konfigurasi kustom maka ganti `{}` dengan parameter baru.

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-version
version-number \
    --service-account-role-arn arn:aws:iam::111122223333:role/role-name --
configuration-values '{}' --resolve-conflicts PRESERVE
```

6. Periksa status pembaruan. Ganti `my-cluster` dengan nama cluster Anda dan `vpc-cni` dengan nama add-on yang Anda perbarui.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni
```

Contoh output adalah sebagai berikut.

```
{
  "addon": {
    "addonName": "vpc-cni",
    "clusterName": "my-cluster",
    "status": "UPDATING",
  }
}
```

Pembaruan selesai ketika statusnya `ACTIVE`.

Verifikasi kompatibilitas versi add-on Amazon EKS dengan kluster

Sebelum Anda membuat add-on Amazon EKS, Anda perlu memverifikasi bahwa versi add-on Amazon EKS kompatibel dengan cluster Anda.

Gunakan [describe-addon-versions API](#) untuk membuat daftar versi add-on EKS yang tersedia, dan versi Kubernetes mana yang didukung oleh setiap versi addon.

1. Verifikasi AWS CLI diinstal dan bekerja dengan. `aws sts get-caller-identity` Jika perintah ini tidak berfungsi, pelajari cara [Memulai dengan AWS CLI](#).
2. Tentukan nama add-on yang ingin Anda ambil informasi kompatibilitas versi, seperti. `amazon-cloudwatch-observability`
3. Tentukan versi Kubernetes dari kluster Anda, seperti. `1.33`
4. Gunakan AWS CLI untuk mengambil versi addon yang kompatibel dengan versi Kubernetes dari kluster Anda.

```
aws eks describe-addon-versions --addon-name amazon-cloudwatch-observability --kubernetes-version 1.33
```

Contoh output adalah sebagai berikut.

```
{
  "addons": [
    {
      "addonName": "amazon-cloudwatch-observability",
      "type": "observability",
      "addonVersions": [
        {
          "addonVersion": "vX.X.X-eksbuild.X",
          "architecture": [
            "amd64",
            "arm64"
          ],
          "computeTypes": [
            "ec2",
            "auto",
            "hybrid"
          ],
          "compatibilities": [
            {
```

```

        "clusterVersion": "1.33",
        "platformVersions": [
            "*"
        ],
        "defaultVersion": true
    }
],
}
]
}

```

Output ini menunjukkan bahwa versi addon vX.X.X-eksbuild.X kompatibel dengan versi cluster Kubernetes. 1.33

Kompatibilitas add-on dengan tipe komputasi

computeTypesBidang dalam describe-addon-versions output menunjukkan kompatibilitas add-on dengan EKS Auto Mode Managed Nodes atau Hybrid Nodes. Add-on yang ditandai auto berfungsi dengan infrastruktur berbasis cloud dan AWS dikelola EKS Auto Mode, sementara yang ditandai hybrid dapat berjalan di node lokal yang terhubung ke bidang kontrol cloud EKS.

Lihat informasi yang lebih lengkap di [the section called “Pertimbangan untuk Mode Otomatis Amazon EKS”](#).

Menghapus add-on Amazon EKS dari kluster

Anda dapat menghapus add-on Amazon EKS dari cluster Anda menggunakan `eksctl`, AWS Management Console, atau AWS CLI.

Saat Anda menghapus add-on Amazon EKS dari kluster:

- Tidak ada downtime untuk fungsionalitas yang disediakan add-on.
- Jika Anda menggunakan Peran IAM untuk Akun Layanan (IRSA) dan add-on memiliki peran IAM yang terkait dengannya, peran IAM tidak akan dihapus.
- Jika Anda menggunakan Identitas Pod, Asosiasi Identitas Pod yang dimiliki oleh add-on akan dihapus. Jika Anda menentukan `--preserve` opsi ke AWS CLI, asosiasi dipertahankan.
- Amazon EKS berhenti mengelola pengaturan untuk add-on.

- Konsol berhenti memberi tahu Anda saat versi baru tersedia.
- Anda tidak dapat memperbarui add-on menggunakan AWS alat apa pun atau APIs.
- Anda dapat memilih untuk meninggalkan perangkat lunak add-on di cluster Anda sehingga Anda dapat mengelolanya sendiri, atau Anda dapat menghapus perangkat lunak add-on dari cluster Anda. Anda hanya harus menghapus perangkat lunak add-on dari cluster Anda jika tidak ada sumber daya di cluster Anda yang bergantung pada fungsionalitas yang disediakan add-on.

Prasyarat

Lengkapi hal-hal berikut sebelum Anda membuat add-on:

- Sebuah klaster Amazon EKS yang sudah ada. Untuk menyebarkan satu, lihat [Memulai](#).
- Periksa apakah add-on Anda memerlukan peran IAM. Untuk informasi selengkapnya, silakan lihat
- Versi 0.210.0 atau yang lebih baru dari alat baris perintah eksctl yang diinstal pada perangkat Anda atau AWS CloudShell. Untuk menginstal atau memperbarui eksctl, lihat [Instalasi](#) di eksctl dokumentasi..

Prosedur

Anda memiliki dua opsi saat menghapus add-on Amazon EKS.

- Pertahankan perangkat lunak add-on di cluster Anda — Opsi ini menghapus pengelolaan Amazon EKS dari pengaturan apa pun. Ini juga menghapus kemampuan Amazon EKS untuk memberi tahu Anda tentang pembaruan dan secara otomatis memperbarui add-on Amazon EKS setelah Anda memulai pembaruan. Namun, ini mempertahankan perangkat lunak add-on di cluster Anda. Opsi ini menjadikan add-on instalasi yang dikelola sendiri, bukan add-on Amazon EKS. Dengan opsi ini, tidak ada downtime untuk add-on.
- Hapus perangkat lunak add-on sepenuhnya dari klaster Anda — Kami menyarankan Anda menghapus add-on Amazon EKS dari klaster Anda hanya jika tidak ada sumber daya di klaster Anda yang bergantung padanya.

Anda dapat menghapus add-on Amazon EKS menggunakan eksctl, AWS Management Console, atau AWS CLI.

Hapus add-on (eksctl)

1. Tentukan add-on saat ini yang diinstal pada cluster Anda. Ganti *my-cluster* dengan nama kluster Anda.

```
eksctl get addon --cluster my-cluster
```

Contoh output adalah sebagai berikut.

NAME	VERSION	STATUS	ISSUES	IAMROLE	UPDATE AVAILABLE
coredns	v1.8.7-eksbuild.2	ACTIVE	0		
kube-proxy	v1.23.7-eksbuild.1	ACTIVE	0		
vpc-cni	v1.10.4-eksbuild.1	ACTIVE	0		
[...]					

Output Anda mungkin terlihat berbeda, tergantung pada add-on dan versi yang Anda miliki di cluster Anda.

2. Hapus add-on. Ganti *my-cluster* dengan nama cluster Anda dan *name-of-addon* dengan nama add-on dikembalikan dalam output dari langkah sebelumnya yang ingin Anda hapus. Jika Anda menghapus *--preserve* opsi, selain Amazon EKS tidak lagi mengelola add-on, perangkat lunak add-on dihapus dari cluster Anda.

```
eksctl delete addon --cluster my-cluster --name name-of-addon --preserve
```

Untuk informasi selengkapnya tentang menghapus add-on, lihat [Menghapus addons dalam dokumentasi](#). eksctl

Hapus add-on (AWS Konsol)

1. Buka [konsol Amazon EKS](#).
2. Pada panel navigasi sebelah kiri, pilih Kluster.
3. Pilih nama cluster yang ingin Anda hapus add-on Amazon EKS.
4. Pilih tab Add-ons.
5. Pilih add-on yang ingin Anda hapus.
6. Pilih Hapus.
7. Dalam Hapus: *name of addon* konfirmasi kotak dialog, lakukan hal berikut:

- a. Jika Anda ingin Amazon EKS berhenti mengelola pengaturan untuk add-on, pilih **Pertahankan** di kluster. Lakukan ini jika Anda ingin mempertahankan perangkat lunak add-on di cluster Anda. Ini agar Anda dapat mengelola semua pengaturan add-on sendiri.
- b. Masukkan nama add-on.
- c. Pilih **Hapus**.

Hapus add-on (AWS CLI)

1. Anda memerlukan versi 0.210.0 atau yang lebih baru dari alat baris perintah eksctl yang diinstal pada perangkat Anda atau AWS CloudShell. Untuk menginstal atau memperbarui eksctl, lihat [Instalasi](#) dalam eksctl dokumentasi.
2. Lihat daftar add-on yang diinstal. Ganti *my-cluster* dengan nama kluster Anda.

```
aws eks list-addons --cluster-name my-cluster
```

Contoh output adalah sebagai berikut.

```
{
  "addons": [
    "coredns",
    "kube-proxy",
    "vpc-cni",
    "name-of-addon"
  ]
}
```

3. Hapus add-on yang diinstal. Ganti *my-cluster* dengan nama cluster Anda dan *name-of-addon* dengan nama add-on yang ingin Anda hapus. Menghapus *--preserve* menghapus perangkat lunak add-on dari cluster Anda.

```
aws eks delete-addon --cluster-name my-cluster --addon-name name-of-addon --preserve
```

Contoh keluaran yang disingkat adalah sebagai berikut.

```
{
  "addon": {
    "addonName": "name-of-add-on",
    "clusterName": "my-cluster",
  }
}
```



```
    "status": "DELETING",  
  }  
}
```

4. Periksa status penghapusan. Ganti *my-cluster* dengan nama cluster Anda dan *name-of-addon* dengan nama add-on yang Anda hapus.

```
aws eks describe-addon --cluster-name my-cluster --addon-name name-of-addon
```

Setelah add-on dihapus, contoh output adalah sebagai berikut.

```
An error occurred (ResourceNotFoundException) when calling the DescribeAddon  
operation: No addon: name-of-addon found in cluster: my-cluster
```

Peran IAM untuk add-on Amazon EKS

Add-on Amazon EKS tertentu memerlukan peran dan izin IAM untuk menelepon. AWS APIs Misalnya, add-on Amazon VPC CNI memanggil tertentu AWS APIs untuk mengonfigurasi sumber daya jaringan di akun Anda. Add-on ini perlu diberikan izin menggunakan IAM. Lebih khusus lagi, akun layanan pod yang menjalankan add-on perlu dikaitkan dengan peran IAM dengan kebijakan IAM tertentu.

Cara yang disarankan untuk memberikan AWS izin ke beban kerja kluster adalah dengan menggunakan fitur Amazon EKS Pod Identities. Anda dapat menggunakan Pod Identity Association untuk memetakan akun layanan add-on ke peran IAM. Jika sebuah pod menggunakan akun layanan yang memiliki asosiasi, Amazon EKS menetapkan variabel lingkungan dalam kontainer pod. Variabel lingkungan mengonfigurasi AWS SDKs, termasuk AWS CLI, untuk menggunakan kredensial Identitas Pod EKS. Untuk informasi selengkapnya, silakan lihat [the section called "Identitas Pod"](#)

Add-on Amazon EKS dapat membantu mengelola siklus hidup asosiasi identitas pod yang sesuai dengan add-on. Misalnya, Anda dapat membuat atau memperbarui add-on Amazon EKS dan asosiasi identitas pod yang diperlukan dalam satu panggilan API. Amazon EKS juga menyediakan API untuk mengambil kebijakan IAM yang disarankan.

1. Konfirmasikan bahwa [agen identitas pod Amazon EKS](#) telah disiapkan di kluster Anda.
2. Tentukan apakah add-on yang ingin Anda instal memerlukan izin IAM menggunakan operasi CLI `describe-addon-versions` AWS. Jika `requiresIamPermissions` benderanya `true`, maka Anda harus menggunakan `describe-addon-configurations` operasi untuk menentukan izin

yang diperlukan oleh addon. Tanggapan tersebut mencakup daftar kebijakan IAM terkelola yang disarankan.

3. Ambil nama Akun Layanan Kubernetes dan kebijakan IAM menggunakan operasi CLI. `describe-addon-configuration` Evaluasi ruang lingkup kebijakan yang disarankan terhadap persyaratan keamanan Anda.
4. Buat peran IAM menggunakan kebijakan izin yang disarankan, dan kebijakan kepercayaan yang diperlukan oleh Pod Identity. Untuk informasi selengkapnya, lihat [the section called “Membuat asosiasi Pod Identity \(AWS Console\)”](#).
5. Buat atau perbarui add-on Amazon EKS menggunakan CLI. Tentukan setidaknya satu asosiasi identitas pod. Asosiasi identitas pod adalah nama akun layanan Kubernetes, dan ARN dari peran IAM.
 - Asosiasi identitas Pod yang dibuat menggunakan add-on APIs dimiliki oleh add-on masing-masing. Jika Anda menghapus add-on, asosiasi identitas pod juga akan dihapus. Anda dapat mencegah penghapusan cascading ini dengan menggunakan `preserve` opsi saat menghapus addon menggunakan CLI AWS atau API. Anda juga dapat langsung memperbarui atau menghapus asosiasi identitas pod jika perlu. Add-on tidak dapat mengasumsikan kepemilikan asosiasi identitas pod yang ada. Anda harus menghapus asosiasi yang ada dan membuatnya kembali menggunakan add-on membuat atau memperbarui operasi.
 - Amazon EKS merekomendasikan penggunaan asosiasi identitas pod untuk mengelola izin IAM untuk add-on. Metode sebelumnya, peran IAM untuk akun layanan (IRSA), masih didukung. Anda dapat menentukan IRSA `serviceAccountRoleArn` dan asosiasi identitas pod untuk add-on. Jika agen identitas pod EKS diinstal pada cluster, `serviceAccountRoleArn` maka akan diabaikan, dan EKS akan menggunakan asosiasi identitas pod yang disediakan. Jika Pod Identity tidak diaktifkan, `serviceAccountRoleArn` maka akan digunakan.
 - Jika Anda memperbarui asosiasi identitas pod untuk add-on yang ada, Amazon EKS akan memulai restart bergulir dari pod add-on.

Ambil informasi IAM tentang add-on Amazon EKS

Sebelum Anda membuat add-on, gunakan AWS CLI untuk menentukan:

- Jika add-on memerlukan izin IAM
- Kebijakan IAM yang disarankan untuk digunakan

Prosedur

1. Tentukan nama add-on yang ingin Anda instal, dan versi Kubernetes dari klaster Anda. Untuk informasi selengkapnya tentang add-on, lihat [the section called “Add-on Amazon EKS”](#).
2. Gunakan AWS CLI untuk menentukan apakah add-on memerlukan izin IAM.

```
aws eks describe-addon-versions \  
--addon-name <addon-name> \  
--kubernetes-version <kubernetes-version>
```

Misalnya:

```
aws eks describe-addon-versions \  
--addon-name aws-ebs-csi-driver \  
--kubernetes-version 1.30
```

Tinjau output sampel berikut. Perhatikan `requiresIamPermissions` itu `true`, dan versi add-on default. Anda perlu menentukan versi add-on saat mengambil kebijakan IAM yang direkomendasikan.

```
{  
  "addons": [  
    {  
      "addonName": "aws-ebs-csi-driver",  
      "type": "storage",  
      "addonVersions": [  
        {  
          "addonVersion": "v1.31.0-eksbuild.1",  
          "architecture": [  
            "amd64",  
            "arm64"  
          ],  
          "compatibilities": [  
            {  
              "clusterVersion": "1.30",  
              "platformVersions": [  
                "*"   
              ],  
              "defaultVersion": true  
            }  
          ],  
          "requiresIamPermissions": true  
        }  
      ],  
      "requiresIamPermissions": true  
    }  
  ],  
  "requiresIamPermissions": true  
}
```

```

        "requiresConfiguration": false,
        "requiresIamPermissions": true
    },
    [...]

```

3. Jika add-on memerlukan izin IAM, gunakan AWS CLI untuk mengambil kebijakan IAM yang direkomendasikan.

```

aws eks describe-addon-configuration \
--query podIdentityConfiguration \
--addon-name <addon-name> \
--addon-version <addon-version>

```

Misalnya:

```

aws eks describe-addon-configuration \
--query podIdentityConfiguration \
--addon-name aws-ebs-csi-driver \
--addon-version v1.31.0-eksbuild.1

```

Tinjau output berikut. Perhatikan `recommendedManagedPolicies`.

```

[
  {
    "serviceAccount": "ebs-csi-controller-sa",
    "recommendedManagedPolicies": [
      "arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy"
    ]
  }
]

```

4. Buat peran IAM dan lampirkan Kebijakan Terkelola yang direkomendasikan. Atau, tinjau kebijakan terkelola dan cakupan izin sesuai kebutuhan. Untuk mengetahui informasi selengkapnya, lihat [the section called “Membuat asosiasi Pod Identity \(AWS Console\)”](#).

Referensi Dukungan Pod Identity

Tabel berikut menunjukkan apakah add-on Amazon EKS tertentu mendukung EKS Pod Identity.

Nama add-on	Dukungan identitas Pod	Versi minimum yang diperlukan
Pengemudi Amazon EBS CSI	Ya	v1.26.0-eksbuild.1
Amazon VPC CNI	Ya	v1.15.5-eksbuild.1
Pengemudi Amazon EFS CSI	Ya	v2.0.5-eksbuild.1
AWS Distro untuk OpenTelemetry	Ya	v0.94.1-eksbuild.1
Mountpoint untuk Driver Amazon S3 CSI	Tidak	N/A
Agen CloudWatch Observabilitas Amazon	Ya	v3.1.0-eksbuild.1

Tabel ini terakhir diperbarui pada 28 Oktober 2024.

Gunakan Identitas Pod untuk menetapkan peran IAM ke add-on Amazon EKS

Add-on Amazon EKS tertentu memerlukan peran dan izin IAM. Sebelum menambahkan update add-on Amazon EKS untuk menggunakan asosiasi Pod Identity, verifikasi peran dan kebijakan yang akan digunakan. Untuk informasi selengkapnya, lihat [the section called “Ambil informasi IAM”](#).

1. Menentukan:

- `cluster-name`— Nama cluster untuk menginstal add-on ke.
- `addon-name`— Nama add-on untuk menginstal.
- `service-account-name`— Nama Akun Layanan Kubernetes yang digunakan oleh add-on.
- `iam-role-arn`— ARN dari peran IAM dengan izin yang cukup untuk add-on. Peran harus memiliki kebijakan kepercayaan yang diperlukan untuk EKS Pod Identity. Untuk informasi selengkapnya, lihat [the section called “Membuat asosiasi Pod Identity \(AWS Console\)”](#).

2. Perbarui add-on menggunakan AWS CLI. Anda juga dapat menentukan asosiasi Pod Identity saat membuat add-on, menggunakan `--pod-identity-associations` sintaks yang sama. Perhatikan bahwa ketika Anda menentukan asosiasi identitas pod saat memperbarui add-on, semua asosiasi identitas pod sebelumnya akan ditimpa.

```
aws eks update-addon --cluster-name <cluster-name> \
--addon-name <addon-name> \
--pod-identity-associations 'serviceAccount=<service-account-name>,roleArn=<role-arn>'
```

Sebagai contoh:

```
aws eks update-addon --cluster-name mycluster \
--addon-name aws-ebs-csi-driver \
--pod-identity-associations 'serviceAccount=ebs-csi-controller-sa,roleArn=arn:aws:iam::123456789012:role/StorageDriver'
```

3. Validasi asosiasi Pod Identity telah dibuat:

```
aws eks list-pod-identity-associations --cluster-name <cluster-name>
```

Jika berhasil, Anda akan melihat output yang serupa dengan yang berikut. Perhatikan OwnerARN dari add-on EKS.

```
{
  "associations": [
    {
      "clusterName": "mycluster",
      "namespace": "kube-system",
      "serviceAccount": "ebs-csi-controller-sa",
      "associationArn": "arn:aws:eks:us-west-2:123456789012:podidentityassociation/mycluster/a-4wvljrezsukshq1bv",
      "associationId": "a-4wvljrezsukshq1bv",
      "ownerArn": "arn:aws:eks:us-west-2:123456789012:addon/mycluster/aws-ebs-csi-driver/9cc7ce8c-2e15-b0a7-f311-426691cd8546"
    }
  ]
}
```

Hapus asosiasi Pod Identity dari add-on Amazon EKS

Hapus asosiasi Pod Identity dari add-on Amazon EKS.

1. Menentukan:

- `cluster-name`- Nama cluster EKS untuk menginstal add-on ke.
- `addon-name`- Nama add-on Amazon EKS untuk diinstal.

2. Perbarui addon untuk menentukan array kosong asosiasi identitas pod.

```
aws eks update-addon --cluster-name <cluster-name> \  
--addon-name <addon-name> \  
--pod-identity-associations "[]"
```

Memecahkan Masalah Identitas Pod untuk add-on EKS

Jika add-on Anda mengalami error saat mencoba operasi AWS API, SDK, atau CLI, konfirmasi hal berikut:

- Agen Identitas Pod dipasang di klaster Anda.
 - Untuk informasi tentang cara menginstal Agen Identitas Pod, lihat [the section called “Mengatur Agen”](#).
- Add-on memiliki asosiasi Pod Identity yang valid.
 - Gunakan AWS CLI untuk mengambil asosiasi untuk nama akun layanan yang digunakan oleh add-on.

```
aws eks list-pod-identity-associations --cluster-name <cluster-name>
```

- Peran IAM memiliki kebijakan kepercayaan yang diperlukan untuk Identitas Pod.
 - Gunakan AWS CLI untuk mengambil kebijakan kepercayaan untuk add-on.

```
aws iam get-role --role-name <role-name> --query Role.AssumeRolePolicyDocument
```

- Peran IAM memiliki izin yang diperlukan untuk add-on.
 - Gunakan AWS CloudTrail untuk meninjau `AccessDenied` atau `UnauthorizedOperation` acara.
- Nama akun layanan dalam asosiasi identitas pod cocok dengan nama akun layanan yang digunakan oleh add-on.
 - Untuk informasi tentang add-on yang tersedia, lihat [the section called “AWS add-on”](#).
- Periksa konfigurasi `MutatingWebhookConfiguration` bernama `pod-identity-webhook`
 - `admissionReviewVersions` dari webhook harus `v1beta1` dan tidak berfungsi dengan `v1`.

Tentukan bidang yang dapat Anda sesuaikan untuk add-on Amazon EKS

Add-on Amazon EKS diinstal ke kluster Anda menggunakan konfigurasi standar dan praktik terbaik. Untuk informasi selengkapnya tentang menambahkan add-on Amazon EKS ke kluster Anda, lihat [the section called “Add-on Amazon EKS”](#).

Anda mungkin ingin menyesuaikan konfigurasi add-on Amazon EKS untuk mengaktifkan fitur-fitur lanjutan. Amazon EKS menggunakan fitur penerapan sisi server Kubernetes untuk mengaktifkan pengelolaan add-on oleh Amazon EKS tanpa menimpa konfigurasi Anda untuk pengaturan yang tidak dikelola oleh Amazon EKS. Untuk informasi selengkapnya, lihat [Server-Side Apply](#) dalam dokumentasi Kubernetes. Untuk mencapai hal ini, Amazon EKS mengelola seperangkat bidang minimum untuk setiap add-on yang diinstalnya. Anda dapat memodifikasi semua bidang yang tidak dikelola oleh Amazon EKS, atau proses bidang kontrol Kubernetes lainnya seperti `kube-controller-manager`, tanpa masalah.

Important

Memodifikasi bidang yang dikelola oleh Amazon EKS mencegah Amazon EKS mengelola add-on dan dapat mengakibatkan perubahan Anda ditimpa saat add-on diperbarui.

Sintaks manajemen lapangan

Ketika Anda melihat detail untuk objek Kubernetes, bidang terkelola dan tidak terkelola akan ditampilkan dalam output. Bidang terkelola dapat berupa salah satu dari jenis berikut:

- Dikelola sepenuhnya - Semua kunci untuk bidang dikelola oleh Amazon EKS. Modifikasi nilai apa pun menyebabkan konflik.
- Dikelola sebagian - Beberapa kunci untuk bidang dikelola oleh Amazon EKS. Hanya modifikasi pada kunci yang dikelola secara eksplisit oleh Amazon EKS yang menyebabkan konflik.

Kedua jenis bidang ditandai dengan `manager: eks`.

Setiap kunci adalah `.<name>`, mewakili bidang itu sendiri, yang selalu memetakan ke set kosong, atau string yang mewakili sub-bidang atau item. Output untuk manajemen lapangan terdiri dari jenis deklarasi berikut:

- `f:<name>`, `<name>` di mana nama bidang dalam daftar.

- **k**: *keys* , di mana *keys* adalah peta bidang item daftar.
- **v**: *value* , di *value* mana nilai format JSON yang tepat dari item daftar.
- **i**: *index* , di *index* mana posisi item dalam daftar.

Bagian output berikut untuk add-on CoreDNS menggambarkan deklarasi sebelumnya:

- Bidang yang dikelola sepenuhnya - Jika bidang terkelola memiliki **f**: (bidang) yang ditentukan, tetapi tidak ada **k**: (kunci), maka seluruh bidang dikelola. Modifikasi pada nilai apa pun di bidang ini menyebabkan konflik.

Pada output berikut, Anda dapat melihat bahwa wadah bernama `coredns` dikelola oleh eks. Sub-bidang `argsimage`, dan `imagePullPolicy` sub-bidang juga dikelola oleh eks. Modifikasi pada nilai apa pun di bidang ini menyebabkan konflik.

```
[...]
f:containers:
  k:{"name":"coredns"}:
    .: {}
  f:args: {}
  f:image: {}
  f:imagePullPolicy: {}
[...]
```

- Bidang yang dikelola sebagian — Jika kunci terkelola memiliki nilai yang ditentukan, kunci yang dideklarasikan dikelola untuk bidang tersebut. Memodifikasi kunci yang ditentukan menyebabkan konflik.

Dalam output berikut, Anda dapat melihat bahwa eks mengelola `config-volume` dan `tmp volume` yang diatur dengan name kunci.

```
[...]
f:volumes:
  k:{"name":"config-volume"}:
    .: {}
  f:configMap:
    f:items: {}
    f:name: {}
  f:name: {}
```

```
k:{"name":"tmp"}:
  .: {}
  f:name: {}
[...]
manager: eks
[...]
```

- Menambahkan kunci ke bidang yang dikelola sebagian — Jika hanya nilai kunci tertentu yang dikelola, Anda dapat menambahkan kunci tambahan dengan aman, seperti argumen, ke bidang tanpa menyebabkan konflik. Jika Anda menambahkan kunci tambahan, pastikan bidang tersebut tidak dikelola terlebih dahulu. Menambahkan atau memodifikasi nilai apa pun yang dikelola menyebabkan konflik.

Pada output berikut, Anda dapat melihat bahwa name kunci dan name bidang dikelola. Menambahkan atau memodifikasi nama kontainer apa pun menyebabkan konflik dengan kunci terkelola ini.

```
[...]
f:containers:
  k:{"name":"coredns"}:
  [...]
    f:name: {}
  [...]
manager: eks
[...]
```

Prosedur

Anda dapat menggunakan `kubectl` untuk melihat bidang mana yang dikelola oleh Amazon EKS untuk add-on Amazon EKS apa pun.

Anda dapat memodifikasi semua bidang yang tidak dikelola oleh Amazon EKS, atau proses bidang kontrol Kubernetes lainnya seperti `kube-controller-manager`, tanpa masalah.

1. Tentukan add-on mana yang ingin Anda periksa. Untuk melihat semua deployments dan DaemonSets diterapkan ke cluster Anda, lihat [the section called “Akses sumber daya kluster”](#).
2. Melihat bidang terkelola untuk add-on dengan menjalankan perintah berikut:

```
kubectl get type/add-on-name -n add-on-namespace -o yaml
```

Misalnya, Anda dapat melihat bidang terkelola untuk add-on CoreDNS dengan perintah berikut.

```
kubectl get deployment/coredns -n kube-system -o yaml
```

Manajemen lapangan tercantum di bagian berikut dalam output yang dikembalikan.

```
[...]
managedFields:
  - apiVersion: apps/v1
    fieldsType: FieldsV1
    fieldsV1:
[...]
```

Note

Jika Anda tidak melihat `managedFields` di output, tambahkan `--show-managed-fields` ke perintah dan jalankan lagi. Versi `kubectl` yang Anda gunakan menentukan apakah bidang terkelola dikembalikan secara default.

Langkah selanjutnya

Sesuaikan bidang yang tidak dimiliki AWS untuk add-on Anda.

Validasi tanda tangan gambar kontainer selama penerapan

Jika Anda menggunakan [AWS Penandatanganan](#) dan ingin memverifikasi gambar kontainer yang ditandatangani pada saat penerapan, Anda dapat menggunakan salah satu solusi berikut:

- [Gatekeeper dan Ratify](#) — Gunakan Gatekeeper sebagai pengontrol penerimaan dan Ratifikasi yang dikonfigurasi dengan plugin AWS Signer sebagai pengait web untuk memvalidasi tanda tangan.
- [Kyverno](#) — Mesin kebijakan Kubernetes yang dikonfigurasi dengan plugin AWS Signer untuk memvalidasi tanda tangan.

 Note

Sebelum memverifikasi tanda tangan gambar kontainer, konfigurasi kebijakan penyimpanan kepercayaan dan kepercayaan [Notasi](#), seperti yang dipersyaratkan oleh pengontrol penerimaan yang Anda pilih.

Mengatur dan memantau sumber daya cluster

Bab ini mencakup topik berikut untuk membantu mengelola kluster Anda. Anda juga dapat melihat informasi tentang [sumber daya Kubernetes](#) Anda dengan file. AWS Management Console

- Dasbor Kubernetes adalah UI berbasis web tujuan umum untuk kluster Kubernetes. Hal ini memungkinkan pengguna untuk mengelola aplikasi yang berjalan di cluster dan memecahkan masalah mereka, serta mengelola cluster itu sendiri. Untuk informasi selengkapnya, lihat Repositori [Dasbor Kubernetes](#). GitHub
- [the section called “Server metrik”](#) – Server Metrik Kubernetes adalah agregator data penggunaan sumber daya di kluster Anda. Ini tidak digunakan secara default di kluster Anda, tetapi digunakan oleh add-on Kubernetes, seperti Dasbor Kubernetes dan. [the section called “Penskala Otomatis Pod Horizontal”](#) Dalam topik ini Anda akan mempelajari cara menginstal Server Metrik.
- [the section called “Menerapkan aplikasi dengan Helm”](#) – Manajer Helm package untuk Kubernetes membantu Anda menginstal dan mengelola aplikasi di kluster Kubernetes Anda. Topik ini membantu Anda menginstal dan menjalankan binari Helm sehingga Anda dapat menginstal dan mengelola grafik menggunakan Helm CLI pada komputer lokal Anda.
- [the section called “Pemberian tag pada sumber daya Anda”](#) – Untuk membantu Anda mengelola sumber daya Amazon EKS, Anda dapat menetapkan metadata Anda sendiri ke setiap sumber daya dalam bentuk tanda. Topik ini menjelaskan tag dan menunjukkan cara membuatnya.
- [the section called “Kuota layanan”](#)— AWS Akun Anda memiliki kuota default, sebelumnya disebut sebagai batas, untuk setiap layanan. AWS Pelajari tentang kuota untuk Amazon EKS dan cara meningkatkannya.

Pantau dan optimalkan biaya kluster Amazon EKS

Pemantauan biaya adalah aspek penting dalam mengelola cluster Kubernetes Anda di Amazon EKS. Dengan mendapatkan visibilitas ke biaya kluster Anda, Anda dapat mengoptimalkan pemanfaatan sumber daya, menetapkan anggaran, dan membuat keputusan berdasarkan data tentang penerapan Anda. Amazon EKS menyediakan dua solusi pemantauan biaya, masing-masing dengan keunggulan uniknya sendiri, untuk membantu Anda melacak dan mengalokasikan biaya Anda secara efektif:

AWS Data alokasi biaya split penagihan untuk Amazon EKS — Fitur asli ini terintegrasi secara mulus dengan Konsol AWS Penagihan, memungkinkan Anda menganalisis dan mengalokasikan biaya menggunakan antarmuka dan alur kerja yang sama yang Anda gunakan untuk layanan lain.

AWS Dengan alokasi biaya terpisah, Anda dapat memperoleh wawasan tentang biaya Kubernetes Anda secara langsung di samping pengeluaran Anda yang lain AWS , sehingga lebih mudah untuk mengoptimalkan biaya secara holistik di seluruh lingkungan Anda. AWS Anda juga dapat memanfaatkan fitur AWS Penagihan yang ada seperti Cost Categories dan Cost Anomaly Detection untuk lebih meningkatkan kemampuan manajemen biaya Anda. Untuk informasi selengkapnya, lihat [Memahami data alokasi biaya terpisah](#) di Panduan Pengguna AWS Penagihan.

Kubecost — Amazon EKS mendukung Kubecost, alat pemantauan biaya Kubernetes. Kubecost menawarkan pendekatan Kubernetes-native yang kaya fitur untuk pemantauan biaya, menyediakan rincian biaya granular oleh sumber daya Kubernetes, rekomendasi pengoptimalan biaya, serta dasbor dan laporan. out-of-the-box Kubecost juga mengambil data harga yang akurat dengan mengintegrasikan dengan Laporan AWS Biaya dan Penggunaan, memastikan Anda mendapatkan tampilan yang tepat dari biaya Amazon EKS Anda. Pelajari cara [menginstal Kubecost](#). Lihat halaman [Kubecost](#) AWS Marketplace untuk informasi tentang mendapatkan langganan Kubecost gratis.

Lihat biaya menurut Pod dalam AWS penagihan dengan alokasi biaya terpisah

Pemantauan biaya menggunakan data alokasi biaya AWS terpisah untuk Amazon EKS

Anda dapat menggunakan data alokasi biaya AWS terpisah untuk Amazon EKS untuk mendapatkan visibilitas biaya granular untuk kluster Amazon EKS Anda. Hal ini memungkinkan Anda untuk menganalisis, mengoptimalkan, dan chargeback biaya dan penggunaan untuk aplikasi Kubernetes Anda. Anda mengalokasikan biaya aplikasi ke unit bisnis dan tim individual berdasarkan EC2 CPU Amazon dan sumber daya memori yang dikonsumsi oleh aplikasi Kubernetes Anda. Data alokasi biaya terpisah untuk Amazon EKS memberikan visibilitas ke dalam biaya per Pod, dan memungkinkan Anda untuk mengumpulkan data biaya per Pod menggunakan namespace, cluster, dan primitif Kubernetes lainnya. Berikut ini adalah contoh primitif Kubernetes yang dapat Anda gunakan untuk menganalisis data alokasi biaya Amazon EKS.

- Nama kluster
- Deployment
- Namespace
- Simpul
- Nama Beban Kerja
- Jenis Beban Kerja

[Tag alokasi biaya yang ditentukan pengguna](#) juga didukung. Untuk informasi selengkapnya tentang penggunaan data alokasi biaya terpisah, lihat [Memahami data alokasi biaya terpisah](#) di Panduan Pengguna AWS Penagihan.

Mengatur Laporan Biaya dan Penggunaan

Anda dapat mengaktifkan Data Alokasi Biaya Pisahkan untuk EKS di Konsol Manajemen Biaya, Antarmuka Baris AWS Perintah, atau AWS SDKs

Gunakan yang berikut ini untuk Data Alokasi Biaya Split:

1. Pilih untuk Membagi Data Alokasi Biaya. Untuk informasi selengkapnya, lihat [Mengaktifkan data alokasi biaya terpisah](#) di Panduan Pengguna Laporan AWS Biaya dan Penggunaan.
2. Sertakan data dalam laporan baru atau yang sudah ada.
3. Lihat laporannya. Anda dapat menggunakan konsol Billing and Cost Management atau melihat file laporan di Amazon Simple Storage Service.

Instal Kubecost

Amazon EKS mendukung Kubecost, yang dapat Anda gunakan untuk memantau biaya yang dipecah berdasarkan sumber daya Kubernetes termasuk Pod, node, namespace, dan label. Topik ini mencakup pemasangan Kubecost, dan mengakses dasbor Kubecost.

Amazon EKS menyediakan bundel Kubecost yang AWS dioptimalkan untuk visibilitas biaya cluster. Anda dapat menggunakan perjanjian AWS dukungan yang ada untuk mendapatkan dukungan. Untuk informasi selengkapnya tentang versi Kubecost yang tersedia, lihat [the section called “Pelajari lebih lanjut tentang Kubecost”](#)

Note

Kubecost v2 memperkenalkan beberapa fitur baru utama. [Pelajari lebih lanjut tentang Kubecost v2.](#)

[Untuk informasi selengkapnya tentang Kubecost, lihat dokumentasi Kubecost dan Pertanyaan yang sering diajukan.](#)

Instal bundel Kubecost yang dioptimalkan Amazon EKS

Anda dapat menggunakan salah satu prosedur berikut untuk menginstal bundel Kubecost Amazon EKS yang dioptimalkan:

- Sebelum memulai, disarankan untuk meninjau [Kubecost - Ikhtisar Arsitektur](#) untuk memahami cara kerja Kubecost di Amazon EKS.
- Jika Anda baru mengenal Amazon EKS, kami sarankan Anda menggunakan Helm untuk instalasi karena ini menyederhanakan instalasi bundel Kubecost Amazon EKS yang dioptimalkan. Untuk informasi selengkapnya, lihat [Menerapkan Kubecost di kluster Amazon EKS menggunakan add-on Amazon EKS](#).
- Untuk menyesuaikan instalasi, Anda dapat mengonfigurasi bundel Kubecost Amazon EKS yang dioptimalkan dengan Helm. Untuk informasi selengkapnya, lihat [Menerapkan Kubecost di kluster Amazon EKS menggunakan Helm](#) dalam dokumentasi Kubecost.

Akses dasbor Kubecost

Setelah penyiapan bundel Kubecost Amazon EKS mengoptimalkan selesai, Anda harus memiliki akses ke dasbor Kubecost. Lihat informasi yang lebih lengkap di [the section called “Akses Dasbor Kubecost”](#).

Akses Dasbor Kubecost

Prasyarat

1. Pastikan status Pod terkait kubecost adalah “Running”.

```
kubectl get pods --namespace kubecost
```

Akses Dasbor Kubecost

1. Di perangkat Anda, aktifkan port-forwarding untuk mengekspos dasbor Kubecost.

```
kubectl port-forward deployment/kubecost-cost-analyzer 9090 --namespace kubecost
```

Atau, Anda dapat menggunakan [AWS Load Balancer Controller](#) untuk mengekspos Kubecost dan menggunakan Amazon Cognito untuk autentikasi, otorisasi, dan manajemen pengguna. Untuk

informasi selengkapnya, lihat [Cara menggunakan Application Load Balancer dan Amazon Cognito untuk mengautentikasi pengguna untuk aplikasi web Kubernetes](#) Anda.

2. Pada perangkat yang sama yang Anda selesaikan langkah sebelumnya, buka browser web dan masukkan alamat berikut.

```
http://localhost:9090
```

Anda melihat halaman Ikhtisar Kubecost di browser Anda. Mungkin diperlukan waktu 5-10 menit (atau lebih) bagi Kubecost untuk mengumpulkan metrik, tergantung pada ukuran klaster Anda. Anda dapat melihat pengeluaran Amazon EKS Anda, termasuk biaya klaster kumulatif, biaya aset Kubernetes terkait, dan pengeluaran agregat bulanan.

3. Untuk melacak biaya di tingkat klaster, beri tag sumber daya Amazon EKS Anda untuk penagihan. Untuk informasi selengkapnya, lihat [the section called “Menandai sumber daya Anda untuk penagihan”](#).
 - Alokasi biaya — Lihat biaya Amazon EKS bulanan dan biaya kumulatif untuk setiap ruang nama Anda dan dimensi lainnya selama tujuh hari terakhir. Ini berguna untuk memahami bagian mana dari aplikasi Anda yang berkontribusi terhadap pengeluaran Amazon EKS.
 - Aset — Lihat biaya aset AWS infrastruktur yang terkait dengan sumber daya Amazon EKS Anda.

Pelajari lebih lanjut tentang Kubecost

Amazon EKS menyediakan bundel Kubecost yang AWS dioptimalkan untuk visibilitas biaya cluster. Amazon EKS mendukung Kubecost, yang dapat Anda gunakan untuk memantau biaya yang dipecah berdasarkan sumber daya Kubernetes termasuk Pod, node, namespace, dan label.

Topik ini mencakup versi Kubecost yang tersedia, dan perbedaan antara tingkatan yang tersedia. EKS mendukung Kubecost Versi 1 dan Versi 2. Setiap versi tersedia dalam tingkatan yang berbeda. Anda dapat menggunakan paket Kubecost Amazon EKS yang dioptimalkan untuk cluster Amazon EKS Anda tanpa biaya tambahan. Anda mungkin dikenakan biaya untuk penggunaan AWS layanan terkait, seperti Amazon Managed Service for Prometheus. Selain itu, Anda dapat menggunakan perjanjian AWS dukungan yang ada untuk mendapatkan dukungan.

Sebagai administrator platform Kubernetes dan pemimpin keuangan, Anda dapat menggunakan Kubecost untuk memvisualisasikan rincian biaya Amazon EKS, mengalokasikan biaya, dan membebaskan kembali unit organisasi seperti tim aplikasi. Anda dapat memberikan data biaya yang

transparan dan akurat kepada tim internal dan unit bisnis Anda berdasarkan AWS tagihan aktual mereka. Selain itu, Anda juga bisa mendapatkan rekomendasi khusus untuk pengoptimalan biaya berdasarkan lingkungan infrastruktur dan pola penggunaan dalam cluster mereka. [Untuk informasi selengkapnya tentang Kubecost, lihat dokumentasi Kubecost.](#)

Apa perbedaan antara bundel kustom Kubecost dan versi gratis Kubecost (juga dikenal sebagai) OpenCost

AWS dan Kubecost berkolaborasi untuk menawarkan versi Kubecost yang disesuaikan. Versi ini mencakup subset fitur komersial tanpa biaya tambahan. Lihat tabel di bawah ini untuk fitur-fitur yang disertakan dalam bundel kustom Kubecost.

Kubecost v2

Apa perbedaan antara Kubecost v1 dan v2?

Kubecost 2.0 adalah upgrade besar dari versi sebelumnya dan mencakup fitur-fitur baru utama termasuk API Backend baru. Perhatikan bahwa [Alokasi](#) dan [Aset](#) sepenuhnya APIs kompatibel ke belakang. [Harap tinjau dokumentasi Kubecost untuk memastikan transisi yang lancar.](#) [Untuk daftar lengkap penyempurnaan, silakan lihat pengumuman Kubecost v2.0 dan catatan rilis lengkapnya.](#)

Important

[Tinjau dokumentasi Kubecost sebelum melakukan upgrade.](#) Peningkatan dapat memengaruhi ketersediaan laporan.

Perbandingan fitur inti:

Fitur	Kubecost tingkat gratis 2.0	Amazon EKS dioptimalkan bundel Kubecost 2.0	Kubecost Perusahaan 2.0
Visibilitas biaya cluster	Cluster tak terbatas hingga 250 core	Multi-cluster terpadu tanpa batas inti saat diintegrasikan dengan Amazon Managed Service untuk Prometheus	Jumlah cluster terpadu dan tidak terbatas di lingkungan yang tidak terbatas (yaitu multi-cloud)

Fitur	Kubecost tingkat gratis 2.0	Amazon EKS dioptimalkan bundel Kubecost 2.0	Kubecost Perusahaan 2.0
Deployment	Pengguna di-host	Pengguna di-host	Dihosting pengguna, Kubecost host (penyewa khusus), SaaS
Database yang didukung	Prometheus Lokal	Layanan Dikelola Amazon untuk Prometheus atau Prometheus Lokal	Setiap rasa prometheus dan basis data khusus
Dukungan retensi database (metrik mentah)	15 hari	Data historis tak terbatas	Data historis tak terbatas
Retensi API dan UI Kubecost (ETL)	15 hari	15 hari	Tidak terbatas.
Visibilitas cloud hybrid	-	Amazon EKS dan Amazon EKS Anywhere cluster	Multi-cloud dan cloud hybrid
Peringatan dan laporan berulang	Hanya didukung pada cluster primer, terbatas pada 250 core	Peringatan efisiensi, peringatan anggaran, peringatan perubahan belanja, dan lebih banyak lagi yang didukung di semua cluster	Peringatan efisiensi, peringatan anggaran, peringatan perubahan belanja, dan lebih banyak lagi yang didukung di semua cluster
Laporan tersimpan	-	Laporan menggunakan metrik 15 hari	Laporan menggunakan data historis dan metrik tak terbatas

Fitur	Kubecost tingkat gratis 2.0	Amazon EKS dioptimalkan bundel Kubecost 2.0	Kubecost Perusahaan 2.0
Integrasi penagihan cloud	Hanya didukung pada cluster primer, terbatas pada 250 core	Dukungan harga khusus untuk AWS (termasuk beberapa cluster dan beberapa akun)	Dukungan harga khusus untuk cloud apa pun
Rekomendasi tabungan	Hanya didukung pada cluster primer, terbatas pada 250 core	Wawasan cluster primer, tetapi tidak ada batas 250 inti	Wawasan multi-cluster
Tata Kelola: Audit	-	-	Audit peristiwa biaya historis
Dukungan Single Sign-on (SSO)	-	Amazon Cognito didukung	Okta, Auth0, PingId KeyCloak, dan apa pun yang kustom
Kontrol akses berbasis peran (RBAC) dengan SAFL 2.0	-	-	Okta, Auth0, PingId KeyCloak, dan apa pun yang kustom
Pelatihan dan orientasi perusahaan	-	-	Pelatihan dan FinOps orientasi layanan lengkap
Tim	-	-	Ya

Fitur Baru:

Fitur-fitur berikut memiliki batas metrik:

- Agregator Kubecost

- Pemantauan Jaringan
- Tindakan Kubecost
- Koleksi
- Deteksi anomali
- Permintaan Kontainer Ukuran Kanan
- Peramalan Kubecost
- Pelengkapan otomatis untuk penyaringan dan agregasi

Batas metrik:

Metrik	Kubecost Gratis Tier 2.0	Amazon EKS dioptimalkan bundel Kubecost 2.0	Kubecost Perusahaan 2.0
Ukuran cluster	Cluster tak terbatas hingga 250 core	Tidak terbatas.	Tidak terbatas.
Retensi metrik	15 hari	15 hari	Tidak terbatas.
Dukungan multi-cluster	Tidak tersedia	Tersedia	Tersedia
Batas inti	250 core per cluster	Tidak ada batas inti	Tidak ada batas inti

Kubecost v1

Fitur	Tingkat gratis Kubecost	Amazon EKS dioptimalkan bundel Kubecost	Perusahaan Kubecost
Deployment	Pengguna di-host	Pengguna di-host	Dihosting pengguna atau di-host Kubecost (SaaS)

Fitur	Tingkat gratis Kubecost	Amazon EKS dioptimalkan bundel Kubecost	Perusahaan Kubecost
Jumlah cluster yang didukung	Tidak terbatas.	Tidak terbatas.	Tidak terbatas.
Database didukung	Prometheus Lokal	Prometheus Lokal atau Layanan Dikelola Amazon untuk Prometheus	Prometheus, Layanan Dikelola Amazon untuk Prometheus, Cortex, atau Thanos
Dukungan retensi basis data	15 hari	Data historis tak terbatas	Data historis tak terbatas
Retensi API Kubecost (ETL)	15 hari	15 hari	Data historis tak terbatas
Visibilitas biaya cluster	Cluster tunggal	Multi-cluster terpadu	Multi-cluster terpadu
Visibilitas cloud hybrid	-	Amazon EKS dan Amazon EKS Anywhere cluster	Dukungan multi-cloud dan hybrid cloud
Peringatan dan laporan berulang	-	Peringatan efisiensi, peringatan anggaran, peringatan perubahan belanja, dan lebih banyak lagi yang didukung	Peringatan efisiensi, peringatan anggaran, peringatan perubahan belanja, dan lebih banyak lagi yang didukung
Laporan tersimpan	-	Laporan menggunakan data 15 hari	Laporan menggunakan data historis tak terbatas

Fitur	Tingkat gratis Kubecost	Amazon EKS dioptimalkan bundel Kubecost	Perusahaan Kubecost
Integrasi penagihan cloud	Diperlukan untuk setiap cluster individu	Dukungan harga khusus untuk AWS (termasuk beberapa cluster dan beberapa akun)	Dukungan harga khusus untuk AWS (termasuk beberapa cluster dan beberapa akun)
Rekomendasi tabungan	Wawasan cluster tunggal	Wawasan cluster tunggal	Wawasan multi-cluster
Tata Kelola: Audit	-	-	Audit peristiwa biaya historis
Dukungan masuk tunggal (SSO)	-	Amazon Cognito didukung	Okta, Auth0, PingID, KeyCloak
Kontrol akses berbasis peran (RBAC) dengan SAFL 2.0	-	-	Okta, Auth0, PingID, Keycloak
Pelatihan dan orientasi perusahaan	-	-	Pelatihan dan FinOps orientasi layanan lengkap

Pertanyaan umum

Lihat pertanyaan dan jawaban umum berikut tentang penggunaan Kubecost dengan Amazon EKS.

Apa itu fitur retensi API Kubecost (ETL)?

Fitur Kubecost ETL menggabungkan dan mengatur metrik ke visibilitas biaya permukaan pada berbagai tingkat granularitas (seperti, dan). namespace-level pod-level deployment-level Untuk paket Kubecost Amazon EKS yang dioptimalkan, pelanggan mendapatkan data dan wawasan dari metrik selama 15 hari terakhir.

Apa fitur peringatan dan laporan berulang? Peringatan dan laporan apa yang disertakan?

Peringatan Kubecost memungkinkan tim menerima pembaruan pada pembelanjaan Kubernetes secara real-time serta pembelanjaan cloud. Laporan berulang memungkinkan tim untuk menerima tampilan kustomisasi dari Kubernetes historis dan belanja cloud. Keduanya dapat dikonfigurasi menggunakan Kubecost UI atau nilai Helm. Mereka mendukung email, Slack, dan Microsoft Teams.

Apa yang termasuk dalam laporan yang disimpan?

Laporan tersimpan Kubecost adalah tampilan metrik biaya dan efisiensi yang telah ditentukan sebelumnya. Mereka termasuk biaya berdasarkan cluster, namespace, label, dan banyak lagi.

Apa itu integrasi penagihan cloud?

Integrasi dengan AWS penagihan APIs memungkinkan Kubecost menampilkan out-of-cluster biaya (seperti Amazon S3). Selain itu, Kubecost memungkinkan Kubecost untuk mendamaikan prediksi in-cluster Kubecost dengan data penagihan aktual untuk memperhitungkan penggunaan spot, paket tabungan, dan diskon perusahaan.

Apa yang termasuk dalam rekomendasi tabungan?

Kubecost memberikan wawasan dan otomatisasi untuk membantu pengguna mengoptimalkan infrastruktur dan pembelanjaan Kubernetes mereka.

Apakah ada biaya untuk fungsi ini?

Tidak. Anda dapat menggunakan paket Kubecost Amazon EKS yang dioptimalkan tanpa biaya tambahan. Jika Anda menginginkan kapabilitas Kubecost tambahan yang tidak disertakan, Anda dapat membeli Lisensi Perusahaan Kubecost melalui AWS Marketplace, atau langsung dari Kubecost.

Apakah dukungan tersedia untuk bundel Kubecost Amazon EKS yang dioptimalkan?

Ya, hanya jika Anda menggunakan bundel Kubecost Amazon EKS yang dioptimalkan.

Bagaimana cara mendapatkan dukungan untuk bundel Kubecost Amazon EKS yang dioptimalkan?

Anda dapat membuka kasus dukungan dengan tim AWS Support di [Contact AWS](#).

Apakah saya memerlukan lisensi untuk menggunakan fitur Kubecost yang disediakan oleh integrasi Amazon EKS?

Tidak.

Dapatkah saya mengintegrasikan Kubecost dengan AWS Cost and Usage Report untuk pelaporan yang lebih akurat?

Ya. Anda dapat mengonfigurasi Kubecost untuk mengambil data dari Laporan AWS Biaya dan Penggunaan untuk mendapatkan visibilitas biaya yang akurat, termasuk diskon, harga Spot, harga instans yang dipesan, dan lainnya. Untuk informasi selengkapnya, lihat [Integrasi Penagihan AWS Cloud](#) di dokumentasi Kubecost.

Apakah versi ini mendukung manajemen biaya cluster Kubernetes yang dikelola sendiri di Amazon? EC2

Tidak. Paket Kubecost yang dioptimalkan Amazon EKS hanya kompatibel dengan kluster Amazon EKS.

Bisakah Kubecost melacak biaya untuk Amazon EKS di Fargate? AWS

Kubecost memberikan upaya terbaik untuk menunjukkan visibilitas biaya cluster untuk Amazon EKS di Fargate, tetapi dengan akurasi yang lebih rendah dibandingkan dengan Amazon EKS di Amazon. EC2 Hal ini terutama disebabkan oleh perbedaan dalam bagaimana Anda ditagih untuk penggunaan Anda. Dengan Amazon EKS di Fargate, Anda ditagih untuk sumber daya yang dikonsumsi. Dengan Amazon EKS di EC2 node Amazon, Anda ditagih untuk sumber daya yang disediakan. Kubecost menghitung biaya EC2 node Amazon berdasarkan spesifikasi node, yang mencakup CPU, RAM, dan penyimpanan sementara. Dengan Fargate, biaya dihitung berdasarkan sumber daya yang diminta untuk Pod Fargate.

Bagaimana saya bisa mendapatkan update dan versi baru dari Kubecost?

Anda dapat meng-upgrade versi Kubecost Anda menggunakan prosedur upgrade Helm standar. Versi terbaru ada di [Galeri Publik Amazon ECR](#).

Apakah **kubect1-cost** CLI didukung? Bagaimana cara menginstalnya?

Ya. **kubect1-cost** adalah alat open source oleh Kubecost (Lisensi Apache 2.0) yang menyediakan akses CLI ke metrik alokasi biaya Kubernetes. Untuk menginstalkubect1-cost, lihat [Instalasi](#) di GitHub.

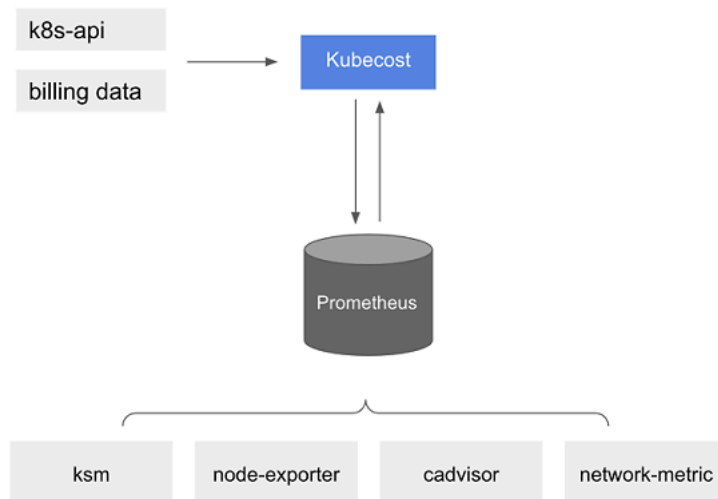
Apakah antarmuka pengguna Kubecost didukung? Bagaimana cara mengaksesnya?

Kubecost menyediakan dasbor web yang dapat Anda akses melalui kubect1 port forwarding, ingress, atau load balancer. Anda juga dapat menggunakan AWS Load Balancer Controller untuk

mengekspos Kubecost dan menggunakan Amazon Cognito untuk autentikasi, otorisasi, dan manajemen pengguna. Untuk informasi selengkapnya, lihat [Cara menggunakan Application Load Balancer dan Amazon Cognito untuk mengautentikasi pengguna untuk aplikasi web Kubernetes Anda](#) di blog. AWS

Fitur Kubecost Tambahan

- Fitur-fitur berikut tersedia di Kubecost v1 dan v2.
 - Metrik biaya ekspor - Pemantauan biaya yang dioptimalkan Amazon EKS diterapkan dengan Kubecost dan Prometheus, yang merupakan sistem pemantauan sumber terbuka dan database deret waktu. Kubecost membaca metrik dari Prometheus dan kemudian melakukan perhitungan alokasi biaya dan menulis metrik kembali ke Prometheus. Front-end Kubecost membaca metrik dari Prometheus dan menampilkannya di antarmuka pengguna Kubecost. Arsitektur diilustrasikan dalam diagram berikut.



Dengan [Prometheus](#) yang sudah diinstal sebelumnya, Anda dapat menulis kueri untuk memasukkan data Kubecost ke dalam sistem intelijen bisnis Anda saat ini untuk analisis lebih lanjut. Anda juga dapat menggunakannya sebagai sumber data untuk dasbor [Grafana](#) Anda saat ini untuk menampilkan biaya kluster Amazon EKS yang akrab dengan tim internal Anda. [Untuk mempelajari lebih lanjut tentang cara menulis kueri Prometheus, lihat <https://opencost.io/docs/installation/prometheus/>](#) readmeberkas pada GitHub atau gunakan contoh model Grafana JSON di repositori [Kubecost Github sebagai referensi](#).

- AWS Integrasi Laporan Biaya dan Penggunaan — Untuk melakukan perhitungan alokasi biaya untuk kluster Amazon EKS Anda, Kubecost mengambil informasi harga publik AWS layanan dan AWS sumber daya dari API Daftar Harga. AWS Anda juga dapat mengintegrasikan Kubecost

dengan Laporan AWS Biaya dan Penggunaan untuk meningkatkan keakuratan informasi harga khusus untuk akun Anda. AWS Informasi ini mencakup program diskon perusahaan, penggunaan instans cadangan, paket tabungan, dan penggunaan spot. Untuk mempelajari lebih lanjut tentang cara kerja integrasi Laporan AWS Biaya dan Penggunaan, lihat [Integrasi Penagihan AWS Cloud dalam dokumentasi](#) Kubecost.

Melihat penggunaan resource dengan Kubernetes Metrics Server

Kubernetes Metrics Server adalah agregator data penggunaan sumber daya di kluster Anda, dan tidak digunakan secara default di kluster Amazon EKS. Untuk informasi selengkapnya, lihat [Kubernetes Metrics Server aktif](#). GitHub [Metrics Server biasanya digunakan oleh add on Kubernetes lainnya, seperti deployment pod Scale dengan Horizontal Pod Autoscaler atau Kubernetes Dashboard](#). Untuk informasi selengkapnya, lihat [Alur metrik sumber daya](#) dalam dokumentasi Kubernetes. Topik ini menjelaskan cara men-deploy Server Metrik Kubernetes di kluster Amazon EKS Anda.

Important

Metrik dimaksudkan untuk point-in-time analisis dan bukan sumber akurat untuk analisis historis. Mereka tidak dapat digunakan sebagai solusi pemantauan atau untuk tujuan penskalaan non-otomatis lainnya. Untuk informasi tentang alat pemantauan, lihat [Monitor cluster](#).

Pertimbangan

- Jika secara manual menerapkan Kubernetes Metrics Server ke node Fargate menggunakan manifes, konfigurasi `metrics-server` deployment untuk menggunakan port selain default. 10250 Pelabuhan ini disediakan untuk Fargate. Versi add-on Amazon EKS dari Metrics Server sudah dikonfigurasikan sebelumnya untuk menggunakan port. 10251
- Pastikan grup keamanan dan jaringan ACLs mengizinkan port 10250 antara `metrics-server` Pod dan semua node dan Pod lainnya. Server Metrik Kubernetes masih menggunakan port 10250 untuk mengumpulkan metrik dari titik akhir lain di cluster. Jika Anda menerapkan pada node Fargate, izinkan port dan port Server Metrik alternatif yang dikonfigurasi. 10250

Terapkan sebagai add-on komunitas dengan Pengaya Amazon EKS

Baru: Anda sekarang dapat menerapkan Metrics Server sebagai add-on komunitas menggunakan konsol AWS atau Amazon EKS. APIs

Terapkan dengan konsol AWS

1. Buka kluster EKS Anda di AWS konsol
2. Dari tab “Add-ons”, pilih Get More Add-ons.
3. Dari bagian “Add-on komunitas”, pilih Server Metrik dan kemudian Berikutnya
4. EKS menentukan versi add-on yang sesuai untuk cluster Anda. Anda dapat mengubah versi menggunakan menu dropdown Versi.
5. Pilih Berikutnya dan kemudian Buat untuk menginstal add-on.

Sumber daya tambahan

Pelajari lebih lanjut tentang [the section called “Pengaya komunitas”](#).

Anda menginstal atau memperbarui add-on komunitas dengan cara yang sama seperti Add-on Amazon EKS lainnya.

- [the section called “Buat add-on”](#)
- [the section called “Perbarui add-on”](#)
- [the section called “Hapus add-on”](#)

Menyebarkan dengan manifes

Baru: Anda sekarang dapat menerapkan Metrics Server sebagai add-on komunitas menggunakan konsol AWS atau Amazon EKS. APIs Petunjuk pemasangan manifes ini akan diarsipkan.

1. Deploy Server Metrik dengan perintah berikut:

```
kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml
```

Jika Anda menggunakan Fargate, Anda harus mengubah file ini. Dalam konfigurasi default, server metrik menggunakan port 10250. Pelabuhan ini disediakan di Fargate. Ganti referensi ke port 10250 di `components.yaml` dengan port lain, seperti 10251.

2. Verifikasi bahwa `metrics-server` deployment menjalankan jumlah Pod yang diinginkan dengan perintah berikut.

```
kubectl get deployment metrics-server -n kube-system
```

Contoh output adalah sebagai berikut.

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
metrics-server	1/1	1	1	6m

3. Uji server metrik bekerja dengan menampilkan sumber daya (CPU/memori) penggunaan node.

```
kubectl top nodes
```

4. Jika Anda menerima pesan kesalahan `Error from server (Forbidden)`, Anda perlu memperbarui konfigurasi Kubernetes RBAC Anda. Identitas Kubernetes RBAC Anda membutuhkan izin yang cukup untuk membaca metrik klaster. Tinjau [izin API Kubernetes minimum yang diperlukan untuk membaca metrik](#). GitHub Pelajari cara [memberikan AWS Identitas IAM seperti Peran akses ke](#) Kubernetes. APIs

Menerapkan aplikasi dengan Helm di Amazon EKS

Manajer paket Helm untuk Kubernetes membantu Anda menginstal dan mengelola aplikasi di klaster Kubernetes Anda. Untuk informasi selengkapnya, lihat [Dokumentasi Helm](#). Topik ini membantu Anda menginstal dan menjalankan biner Helm sehingga Anda dapat menginstal dan mengelola grafik menggunakan Helm CLI pada sistem lokal Anda.

Important

Sebelum Anda dapat menginstal grafik Helm di Amazon EKS klaster Anda, Anda harus konfigurasi `kubectl` untuk bekerja untuk Amazon EKS. Jika Anda belum melakukannya, lihat [the section called “Akses klaster dengan kubectl”](#) sebelum melanjutkan. Jika perintah berikut berhasil untuk cluster Anda, Anda dikonfigurasi dengan benar.

```
kubectl get svc
```

1. Jalankan perintah yang sesuai untuk sistem operasi klien Anda.

- Jika Anda menggunakan macOS dengan [Homebrew](#), instal binari dengan perintah berikut.

```
brew install helm
```

- Untuk opsi penginstalan lainnya, lihat [Menginstal Helm](#) di Helm Docs.

 Note

Jika Anda mendapatkan pesan yang openssl harus diinstal terlebih dahulu, Anda dapat menginstalnya dengan perintah berikut.

```
sudo yum install openssl
```

1. Untuk mengambil biner baru di PATH, Tutup jendela terminal Anda saat ini dan buka jendela baru.
2. Lihat versi Helm yang Anda instal.

```
helm version --template='{{ .Version }}{{ "\n" }}'
```

Contoh output adalah sebagai berikut.

```
v3.17.2
```

3. Pastikan versi yang diinstal kompatibel dengan versi cluster Anda. Periksa [Versi yang Didukung Skew](#) untuk mempelajari lebih lanjut. Misalnya, jika Anda menjalankan 3.17.x, versi Kubernetes yang didukung tidak boleh keluar dari kisaran ~.1.29.x 1.32.x
4. Pada titik ini, Anda dapat menjalankan perintah Helm apa pun (seperti `helm install chart-name`) untuk menginstal, memodifikasi, menghapus, atau kueri Helm bagan di klaster Anda. Jika Anda baru mengenal Helm dan tidak memiliki bagan khusus untuk dipasang, Anda dapat:
 - Percobaan dengan memasang contoh bagan. Melihat [Memasang contoh bagan](#) di Helm [Panduan Quick Start](#).

- Membuat contoh grafik dan mengajukannya ke Amazon ECR. Untuk informasi selengkapnya, lihat [Mengajukan grafik Helm](#) di Panduan Pengguna Registri Kontainer Elastis Amazon.
- Instal bagan Amazon EKS dari GitHub repo [eks-charts](#) atau dari [ArtifactHub](#)

Mengatur sumber daya Amazon EKS dengan tag

Anda dapat menggunakan tag untuk membantu mengelola sumber daya Amazon EKS Anda. Topik ini memberikan gambaran umum pada fungsi tanda serta menunjukkan bagaimana Anda bisa membuat tanda.

Topik

- [Dasar-dasar tag](#)
- [Pemberian tag pada sumber daya Anda](#)
- [Batasan tag](#)
- [Menandai sumber daya Anda untuk penagihan](#)
- [Bekerja dengan tanda menggunakan konsol](#)
- [Cara menggunakan tanda dengan menggunakan CLI, API, atau eksctl](#)

Note

Tag adalah jenis metadata yang terpisah dari label dan anotasi Kubernetes. Untuk informasi selengkapnya tentang jenis metadata lainnya, lihat bagian berikut dalam dokumentasi Kubernetes:

- [Label dan Selektor](#)
- [Anotasi](#)

Dasar-dasar tag

Tag adalah label yang Anda tetapkan ke AWS sumber daya. Setiap tag terdiri dari kunci dan nilai opsional.

Dengan tag, Anda dapat mengkategorikan sumber daya Anda AWS . Misalnya, Anda dapat mengkategorikan sumber daya berdasarkan tujuan, pemilik, atau lingkungan. Bila Anda memiliki

banyak sumber daya dari jenis yang sama, Anda dapat menggunakan tag yang Anda tetapkan ke sumber daya tertentu untuk mengidentifikasi sumber daya tersebut dengan cepat. Misalnya, Anda dapat menentukan satu set tag untuk kluster Amazon EKS Anda untuk membantu Anda melacak setiap pemilik kluster dan tingkat tumpukan. Kami menyarankan agar Anda merancang serangkaian konsisten kunci tanda untuk setiap jenis sumber daya. Kemudian Anda dapat mencari dan memfilter sumber daya berdasarkan tanda yang Anda tambahkan.

Setelah Anda menambahkan sebuah tanda, Anda dapat mengedit kunci serta nilai tanda atau menghilangkan tanda dari sumber daya kapanpun yang Anda mau. Jika Anda menghapus sebuah sumber daya, tanda apapun untuk sumber daya tersebut juga dihapus.

Tag tidak memiliki arti semantik untuk Amazon EKS dan ditafsirkan secara ketat sebagai serangkaian karakter. Anda dapat mengatur nilai tag ke string kosong. Namun, Anda tidak dapat mengatur nilai tag ke null. Jika Anda menambahkan tanda yang memiliki kunci yang sama dengan tanda yang ada pada sumber daya tersebut, nilai baru akan menimpa nilai sebelumnya.

Jika Anda menggunakan AWS Identity and Access Management (IAM), Anda dapat mengontrol pengguna mana di AWS akun Anda yang memiliki izin untuk mengelola tag.

Pemberian tag pada sumber daya Anda

Tag dukungan sumber daya Amazon EKS berikut:

- kluster
- kelompok simpul terkelola
- Profil Fargate

Anda dapat menandai sumber daya ini menggunakan yang berikut:

- Jika Anda menggunakan konsol Amazon EKS, Anda dapat menerapkan tag ke sumber daya baru atau yang sudah ada kapan saja. Anda dapat melakukannya dengan menggunakan tabel Tanda pada halaman sumber daya yang relevan. Untuk informasi selengkapnya, lihat [the section called “Bekerja dengan tanda menggunakan konsol”](#).
- Jika Anda menggunakan `eksctl`, Anda dapat menerapkan tag ke sumber daya saat dibuat menggunakan `--tags` opsi.
- Jika Anda menggunakan AWS CLI, Amazon EKS API, atau AWS SDK, Anda dapat menerapkan tag ke sumber daya baru menggunakan `tags` parameter pada tindakan API yang relevan. Anda

dapat memasang tanda ke sumber daya yang ada menggunakan tindakan API `TagResource`. Untuk informasi selengkapnya, lihat [TagResource](#).

Saat Anda menggunakan beberapa tindakan pembuatan sumber daya, Anda juga dapat menentukan tag untuk sumber daya pada saat yang sama saat Anda membuatnya. Jika tag tidak dapat diterapkan saat sumber daya sedang dibuat, sumber daya gagal dibuat. Mekanisme ini memastikan bahwa sumber daya yang ingin Anda tag dibuat dengan tag yang Anda tentukan atau tidak dibuat sama sekali. Jika Anda menandai sumber daya saat membuatnya, Anda tidak perlu menjalankan skrip penandaan khusus setelah membuat sumber daya.

Tag tidak menyebar ke sumber daya lain yang terkait dengan sumber daya yang Anda buat. Misalnya, tag profil Fargate tidak menyebar ke sumber daya lain yang terkait dengan profil Fargate, seperti Pod yang dijadwalkan dengannya.

Batasan tag

Pembatasan berikut berlaku untuk tag:

- Maksimal 50 tag dapat dikaitkan dengan sumber daya.
- Kunci tag tidak dapat diulang untuk satu sumber daya. Setiap kunci tag harus unik, dan hanya dapat memiliki satu nilai.
- Panjang tombol dapat mencapai 128 karakter di UTF-8.
- Nilai dapat mencapai 256 karakter di UTF-8.
- Jika beberapa AWS layanan dan sumber daya menggunakan skema penandaan Anda, batasi jenis karakter yang Anda gunakan. Beberapa layanan mungkin memiliki batasan pada karakter yang diizinkan. Umumnya, karakter yang diizinkan adalah huruf, angka, spasi, dan karakter berikut: + - = . _ : /@.
- Kunci dan nilai tanda peka huruf besar-kecil.
- Jangan gunakan `aws :`, `AWS :`, atau kombinasi huruf besar atau kecil seperti awalan untuk kunci atau nilai. Ini hanya disediakan untuk AWS digunakan. Anda tidak dapat mengedit atau menghapus kunci tag atau nilai dengan awalan ini. Tag dengan awalan ini tidak dihitung terhadap tags-per-resource batas Anda.

Menandai sumber daya Anda untuk penagihan

Saat Anda menerapkan tag ke kluster Amazon EKS, Anda dapat menggunakannya untuk alokasi biaya di Laporan Biaya & Penggunaan Anda. Data pengukuran dalam Laporan Biaya & Penggunaan Anda menunjukkan penggunaan di semua kluster Amazon EKS Anda. Untuk informasi selengkapnya, lihat [laporan AWS biaya dan penggunaan](#) di Panduan Pengguna AWS Penagihan.

Tag alokasi biaya yang AWS dihasilkan, khususnya `aws:eks:cluster-name`, memungkinkan Anda memecah biaya EC2 instans Amazon berdasarkan kluster Amazon EKS individual di Cost Explorer. Namun, tag ini tidak menangkap biaya pesawat kontrol. Tag secara otomatis ditambahkan ke EC2 instans Amazon yang berpartisipasi dalam kluster Amazon EKS. Perilaku ini terjadi terlepas dari apakah instance disediakan menggunakan grup simpul terkelola Amazon EKS, Karpenter, atau langsung dengan Amazon. EC2 Tag khusus ini tidak dihitung terhadap batas 50 tag. Untuk menggunakan tag, pemilik akun harus mengaktifkannya di konsol AWS Penagihan atau dengan menggunakan API. Saat pemilik akun manajemen AWS Organisasi mengaktifkan tag, tag tersebut juga diaktifkan untuk semua akun anggota organisasi.

Anda juga dapat mengatur informasi penagihan berdasarkan sumber daya yang memiliki nilai kunci tag yang sama. Misalnya, Anda dapat menandai beberapa sumber daya dengan nama aplikasi tertentu, dan kemudian mengatur informasi penagihan Anda. Dengan begitu, Anda dapat melihat total biaya aplikasi itu di beberapa layanan. Untuk informasi selengkapnya tentang menyiapkan laporan alokasi biaya dengan tag, lihat Laporan [Alokasi Biaya Bulanan di Panduan](#) Pengguna AWS Penagihan.

Note

Jika Anda baru saja mengaktifkan pelaporan, data untuk bulan berjalan tersedia untuk dilihat setelah 24 jam.

Cost Explorer adalah alat pelaporan yang tersedia sebagai bagian dari Tingkat AWS Gratis. Anda dapat menggunakan Cost Explorer untuk melihat bagan sumber daya Amazon EKS Anda dari 13 bulan terakhir. Anda juga dapat memperkirakan berapa banyak yang akan Anda belanjakan untuk tiga bulan ke depan. Anda dapat melihat pola berapa banyak yang Anda habiskan untuk AWS sumber daya dari waktu ke waktu. Misalnya, Anda dapat menggunakannya untuk mengidentifikasi area yang memerlukan penyelidikan lebih lanjut dan melihat tren yang dapat Anda gunakan untuk memahami biaya Anda. Anda juga dapat menentukan rentang waktu untuk data, dan melihat data waktu menurut hari atau bulan.

Bekerja dengan tanda menggunakan konsol

Menggunakan konsol Amazon EKS, Anda dapat mengelola tag yang terkait dengan cluster baru atau yang sudah ada dan grup node terkelola.

Saat Anda memilih halaman khusus sumber daya di konsol Amazon EKS, halaman tersebut menampilkan daftar sumber daya tersebut. Misalnya, jika Anda memilih Cluster dari panel navigasi kiri, konsol akan menampilkan daftar kluster Amazon EKS. Saat Anda memilih sumber daya dari salah satu daftar (misalnya, kluster tertentu) yang mendukung tanda ini, Anda dapat melihat dan mengelola tandanya di tab Tanda.

Anda juga dapat menggunakan Editor Tag di AWS Management Console, yang menyediakan cara terpadu untuk mengelola tag Anda. Untuk informasi selengkapnya, lihat [Menandai AWS sumber daya Anda dengan Editor AWS Tag](#) di Panduan Pengguna Editor Tag.

Menambahkan tag pada sumber daya pada pembuatan

Anda dapat menambahkan tanda ke kluster Amazon EKS, grup simpul terkelola, dan profil Fargate ketika Anda membuat fitur tersebut. Untuk informasi selengkapnya, lihat [the section called “Membuat kluster”](#).

Menambahkan dan menghapus tag pada sumber daya

Anda dapat menambahkan atau menghapus tag yang terkait dengan cluster Anda langsung dari halaman sumber daya.

1. Buka [konsol Amazon EKS](#).
2. Pada bilah navigasi, pilih AWS Wilayah yang akan digunakan.
3. Pada panel navigasi sebelah kiri, pilih Kluster.
4. Pilih kluster tertentu.
5. Pilih tab Tag, lalu pilih Kelola tag.
6. Pada halaman Kelola tag, tambahkan atau hapus tag Anda seperlunya.
 - Untuk menambahkan tanda, pilih Tambahkan tanda. Kemudian tentukan kunci dan nilai untuk setiap tag.
 - Untuk menghapus tag, pilih Hapus tag.
7. Ulangi proses ini untuk setiap tag yang ingin Anda tambahkan atau hapus.
8. Pilih Perbarui untuk menyelesaikan.

Cara menggunakan tanda dengan menggunakan CLI, API, atau eksctl

Gunakan perintah AWS CLI berikut atau operasi Amazon EKS API untuk menambahkan, memperbarui, membuat daftar, dan menghapus tag untuk sumber daya Anda. Anda hanya dapat menggunakan eksctl untuk menambahkan tag sekaligus membuat sumber daya baru dengan satu perintah.

Tugas	AWS CLI	AWS Alat untuk Windows PowerShell	Tindakan API
Tambahkan atau timpa satu atau beberapa tanda.	tag-sumber daya	Tambah- EKSResource Tag	TagResource
Hapus satu atau beberapa tanda.	untag-sumber daya	Hapus- EKSResource Tag	UntagResource

Contoh berikut menunjukkan cara menandai atau menghapus tag sumber daya menggunakan AWS CLI.

Contoh 1: Tandai klaster yang sudah ada

Perintah berikut akan menandai klaster yang sudah ada.

```
aws eks tag-resource --resource-arn resource_ARN --tags team=devs
```

Contoh 2: Hapus tanda dari klaster yang sudah ada

Perintah berikut akan menghapus tanda dari klaster yang sudah ada.

```
aws eks untag-resource --resource-arn resource_ARN --tag-keys tag_key
```

Contoh 3: Cantumkan tanda untuk sumber daya

Perintah berikut mencantumkan tag yang terkait dengan sumber daya yang ada.

```
aws eks list-tags-for-resource --resource-arn resource_ARN
```

Saat Anda menggunakan beberapa tindakan pembuatan sumber daya, Anda dapat menentukan tag pada saat yang sama saat Anda membuat sumber daya. Tindakan berikut mendukung penentuan tag saat Anda membuat sumber daya.

Tugas	AWS CLI	AWS Alat untuk Windows PowerShell	Tindakan API	eksctl
Membuat klaster	buat-kluster	Baru- EKSCluster	CreateCluster	<code>create cluster</code>
Buat grup pengelola simpul*	buat-nodegroup	Baru- EKSNodegroup	CreateNodegroup	<code>create nodegroup</code>
Buat profil Fargate	create-fargate-profile	Baru- EKSFargate Profil	CreateFargateProfile.html	<code>create fargateprofile</code>

- Jika Anda ingin juga menandai EC2 instance Amazon saat membuat grup node terkelola, buat grup node terkelola menggunakan templat peluncuran. Untuk informasi selengkapnya, lihat [the section called “Menandai instans Amazon EC2”](#). Jika instances sudah ada, Anda dapat menambahkan tanda pada instans secara manual. Untuk informasi selengkapnya, lihat [Menandai sumber daya Anda](#) di Panduan EC2 Pengguna Amazon.

Lihat dan kelola kuota layanan Amazon EKS dan Fargate

Amazon EKS telah terintegrasi dengan Service Quotas, AWS layanan yang dapat Anda gunakan untuk melihat dan mengelola kuota Anda dari lokasi pusat. Untuk informasi selengkapnya, lihat [Apa itu Service Quotas?](#) di Panduan Pengguna Service Quotas. Dengan integrasi Service Quotas, Anda dapat dengan cepat mencari nilai kuota layanan Amazon EKS dan AWS Fargate Anda menggunakan CLI. AWS Management Console AWS

Lihat kuota layanan EKS di AWS Management Console

1. Buka [Konsol Service Quotas](#).

2. Di panel navigasi kiri, pilih AWS layanan.
3. Dari daftar AWS layanan, cari dan pilih Amazon Elastic Kubernetes Service (Amazon EKS) atau Fargate. AWS

Dalam daftar Kuota layanan, Anda dapat melihat nama kuota layanan, nilai yang diterapkan (jika tersedia), kuota AWS default, dan apakah nilai kuota dapat disesuaikan.

4. Untuk melihat informasi tambahan tentang service quotas, seperti deskripsi, pilih nama kuota.
5. (Opsional) Untuk meminta peningkatan kuota, pilih kuota yang ingin Anda tingkatkan, kemudian pilih Meminta peningkatan kuota, masukkan atau pilih informasi yang diperlukan, dan pilih Minta.

Untuk bekerja lebih lanjut dengan kuota layanan menggunakan AWS Management Console, lihat Panduan Pengguna [Service Quotas](#). Untuk meminta peningkatan kuota, lihat [Meminta Peningkatan Kuota](#) dalam Panduan Pengguna Service Quotas.

Lihat kuota layanan EKS dengan CLI AWS

Jalankan perintah berikut untuk menampilkan kuota Amazon EKS Anda.

```
aws service-quotas list-aws-default-service-quotas \
  --query 'Quotas[*]'.
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
  --service-code eks \
  --output table
```

Jalankan perintah berikut untuk menampilkan kuota Fargate Anda.

```
aws service-quotas list-aws-default-service-quotas \
  --query 'Quotas[*]'.
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
  --service-code fargate \
  --output table
```

Note

Kuota yang dikembalikan adalah jumlah tugas Amazon ECS atau Pod Amazon EKS yang dapat berjalan secara bersamaan di Fargate di akun ini di Wilayah saat ini. AWS

Untuk bekerja lebih banyak dengan kuota layanan menggunakan CLI, [lihat AWS](#) kuota layanan di AWS Referensi Perintah CLI. Untuk meminta peningkatan kuota, lihat [request-service-quota-increase](#) perintah di Referensi Perintah AWS CLI.

Amazon EKS service quotas

AWS merekomendasikan menggunakan AWS Management Console untuk melihat kuota Anda saat ini. Untuk informasi selengkapnya, lihat [the section called “Lihat kuota layanan EKS di AWS Management Console”](#).

Untuk melihat kuota layanan EKS default, lihat titik [akhir dan kuota Amazon Elastic Kubernetes Service](#) di Referensi Umum. AWS

Kuota layanan ini tercantum di bawah Amazon Elastic Kubernetes Service (Amazon EKS) di konsol Service Quotas. Untuk meminta peningkatan kuota untuk nilai yang ditampilkan sebagai dapat disesuaikan, lihat [Meminta peningkatan kuota dalam Panduan Pengguna Service Quotas](#).

Note

Penyesuaian komponen berikut tidak didukung dalam Service Quotas: * Asosiasi Identitas Pod per kluster. Untuk batas, lihat [the section called “Identitas Pod”](#). * CIDRs untuk Remote Node Networks atau Remote Pod Networks untuk node hybrid. Untuk batas, lihat [the section called “Node hibrida”](#).

AWS Kuota layanan Fargate

Layanan AWS Fargate di konsol Service Quotas mencantumkan beberapa kuota layanan. Anda dapat mengonfigurasi alarm yang memberi tahu Anda saat penggunaan mendekati kuota layanan. Untuk informasi selengkapnya, lihat [the section called “Membuat CloudWatch alarm untuk memantau metrik penggunaan sumber daya Fargate”](#).

AWS Akun baru mungkin memiliki kuota awal yang lebih rendah yang dapat meningkat seiring waktu. Fargate terus memantau penggunaan akun di setiap AWS Wilayah, dan kemudian secara otomatis meningkatkan kuota berdasarkan penggunaan. Anda juga dapat meminta peningkatan kuota untuk nilai yang ditampilkan sebagai dapat disesuaikan. Untuk informasi selengkapnya, lihat [Meminta peningkatan kuota](#) di Panduan Pengguna Service Quotas.

AWS merekomendasikan menggunakan AWS Management Console untuk melihat kuota Anda saat ini. Untuk informasi selengkapnya, lihat [the section called “Lihat kuota layanan EKS di AWS Management Console”](#).

Untuk melihat AWS Fargate default pada kuota layanan EKS, lihat Kuota layanan [Fargate di Referensi Umum](#). AWS

 Note

Fargate juga memberlakukan tugas Amazon ECS dan kuota tingkat peluncuran Amazon EKS Pods. Untuk informasi selengkapnya, lihat [AWS Kuota pembatasan Fargate di panduan Amazon ECS](#).

Keamanan di Amazon EKS

Keamanan cloud di AWS adalah prioritas tertinggi. Sebagai AWS pelanggan, Anda mendapat manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan.

Keamanan adalah tanggung jawab bersama antara Anda AWS dan Anda. [Model tanggung jawab bersama](#) menjelaskan hal ini sebagai keamanan cloud dan keamanan dalam cloud:

- Keamanan cloud — AWS bertanggung jawab untuk melindungi infrastruktur yang menjalankan AWS layanan di AWS Cloud. Untuk Amazon EKS, AWS bertanggung jawab atas bidang kontrol Kubernetes, yang mencakup node dan database bidang kontrol. etcd Auditor pihak ketiga secara teratur menguji dan memverifikasi keefektifan keamanan kami sebagai bagian dari [program kepatuhan AWS](#). Untuk mempelajari tentang program kepatuhan yang berlaku untuk Amazon EKS, lihat [Layanan AWS yang Dicakup oleh Program Kepatuhan](#).
- Keamanan di dalam cloud – Tanggung jawab Anda meliputi area berikut.
 - Konfigurasi keamanan bidang data, mencakup konfigurasi grup keamanan yang memungkinkan lalu lintas berpindah dari bidang kendali Amazon EKS ke VPC pelanggan
 - Konfigurasi simpul dan kontainer itu sendiri
 - Sistem operasi node (termasuk pembaruan dan patch keamanan)
 - Perangkat lunak aplikasi terkait lainnya:
 - Menyiapkan dan mengelola kendali jaringan, misalnya aturan firewall
 - Mengelola identitas tingkat platform dan manajemen akses, baik dengan ataupun selain IAM
 - Sensitivitas data Anda, persyaratan perusahaan Anda, serta hukum dan peraturan yang berlaku

Amazon EKS disertifikasi oleh beberapa program kepatuhan untuk aplikasi yang diatur dan sensitif. [Amazon EKS sesuai dengan SOC, PCI, ISO, FedRAMP-moderat, IRAP, C5, K-ISMS, ENS High, OSPAR, HITRUST CSF, dan merupakan layanan yang memenuhi syarat HIPAA](#). Untuk informasi selengkapnya, lihat [Kelola akses](#).

Dokumentasi ini membantu Anda memahami cara menerapkan model tanggung jawab bersama saat menggunakan Amazon EKS. Topik berikut menunjukkan cara mengonfigurasi Amazon EKS untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga mempelajari cara menggunakan AWS layanan lain yang membantu Anda memantau dan mengamankan sumber daya Amazon EKS Anda.

Note

Kontainer Linux terdiri dari grup kontrol (cgroups) dan ruang nama yang membantu membatasi apa yang dapat diakses oleh wadah, tetapi semua kontainer berbagi kernel Linux yang sama dengan instance Amazon host. EC2 Menjalankan kontainer sebagai pengguna asal (UID 0) atau memberikan akses kontainer ke sumber daya host atau namespaces, seperti jaringan host atau namespace host PID, sangat tidak disarankan karena hal itu dapat mengurangi efektivitas isolasi yang disediakan oleh kontainer.

Topik

- [Amankan kluster Amazon EKS dengan praktik terbaik](#)
- [Analisis kerentanan di Amazon EKS](#)
- [Validasi kepatuhan untuk kluster Amazon EKS](#)
- [Pertimbangan keamanan untuk Amazon Elastic Kubernetes Service](#)
- [Pertimbangan keamanan untuk Kubernetes](#)
- [Pertimbangan keamanan untuk Amazon EKS Auto Mode](#)
- [Identity and access management untuk Amazon EKS](#)

Amankan kluster Amazon EKS dengan praktik terbaik

Praktik terbaik keamanan Amazon EKS ada dalam [Praktik Terbaik untuk Keamanan](#) di Panduan Praktik Terbaik Amazon EKS.

Analisis kerentanan di Amazon EKS

Keamanan adalah pertimbangan penting untuk mengonfigurasi dan memelihara kluster serta aplikasi Kubernetes. Berikut ini mencantumkan sumber daya bagi Anda untuk menganalisis konfigurasi keamanan kluster EKS Anda, sumber daya bagi Anda untuk memeriksa kerentanan, dan integrasi dengan AWS layanan yang dapat melakukan analisis itu untuk Anda.

Patokan Pusat Keamanan Internet (CIS) untuk Amazon EKS

[Pusat Keamanan Internet \(CIS\) Kubernetes Benchmark memberikan](#) panduan untuk konfigurasi keamanan Amazon EKS. Tolok ukur:

- Berlaku untuk EC2 node Amazon (baik yang dikelola maupun yang dikelola sendiri) di mana Anda bertanggung jawab atas konfigurasi keamanan komponen Kubernetes.
- Menyediakan cara standar yang disetujui komunitas untuk memastikan bahwa Anda telah mengonfigurasi klaster dan simpul Kubernetes dengan aman saat menggunakan Amazon EKS.
- Terdiri dari empat bagian; konfigurasi pencatatan bidang kendali, konfigurasi keamanan simpul, kebijakan, dan layanan terkelola.
- Mendukung semua versi Kubernetes yang saat ini tersedia di Amazon EKS dan dapat dijalankan menggunakan [kube-bench](#), yaitu alat sumber terbuka standar untuk memeriksa konfigurasi menggunakan tolok ukur CIS pada klaster Kubernetes.

Untuk mempelajari selengkapnya, lihat [Memperkenalkan Tolok Ukur CIS Amazon EKS](#).

Untuk `aws-sample` pipeline otomatis untuk memperbarui grup node Anda dengan AMI benchmark CIS, lihat AMI Hardening Pipeline yang [dioptimalkan EKS](#).

Amazon EKS versi platform

Versi platform Amazon EKS mewakili kemampuan bidang kontrol cluster, termasuk flag server API Kubernetes yang diaktifkan dan versi patch Kubernetes saat ini. Klaster baru di-deploy dengan versi platform terbaru. Untuk detailnya, lihat [the section called “Versi platform”](#).

Anda dapat [memperbarui klaster Amazon EKS](#) ke versi Kubernetes yang lebih baru. Ketika versi Kubernetes baru tersedia di Amazon EKS, kami sarankan supaya Anda secara proaktif memperbarui klaster Anda untuk menggunakan versi terbaru yang tersedia. Untuk informasi selengkapnya tentang versi Kubernetes di EKS, lihat [the section called “Versi Kubernetes”](#).

Daftar kerentanan sistem operasi

AL2023 daftar kerentanan

Lacak peristiwa keamanan atau privasi untuk Amazon Linux 2023 di [Pusat Keamanan Amazon Linux](#) atau berlangganan umpan [RSS](#) terkait. Kejadian keamanan dan privasi mencakup gambaran umum masalah yang terpengaruh, paket, dan petunjuk untuk memperbarui instans Anda guna memperbaiki masalah.

Daftar kerentanan Amazon Linux 2

Lacak kejadian keamanan atau privasi untuk Amazon Linux 2 di [Pusat Keamanan Amazon Linux](#) atau berlangganan ke [umpan RSS](#) terkait. Kejadian keamanan dan privasi mencakup gambaran

umum masalah yang terpengaruh, paket, dan petunjuk untuk memperbarui instans Anda guna memperbaiki masalah.

Deteksi node dengan Amazon Inspector

Anda dapat menggunakan [Amazon Inspector](#) untuk memeriksa aksesibilitas jaringan node yang tidak diinginkan dan kerentanan pada instans Amazon tersebut. EC2

Deteksi cluster dan node dengan Amazon GuardDuty

Layanan deteksi GuardDuty ancaman Amazon yang membantu melindungi akun, wadah, beban kerja, dan data di AWS lingkungan Anda. Di antara fitur-fitur lainnya, GuardDuty menawarkan dua fitur berikut yang mendeteksi potensi ancaman terhadap kluster EKS Anda: Perlindungan EKS dan Pemantauan Runtime.

Untuk informasi selengkapnya, lihat [the section called “Amazon GuardDuty”](#).

Validasi kepatuhan untuk kluster Amazon EKS

Untuk mengetahui apakah suatu AWS layanan berada dalam lingkup program kepatuhan tertentu, lihat [AWS layanan dalam Lingkup oleh Program Kepatuhan](#) dan pilih program kepatuhan yang Anda minati. Untuk informasi umum, lihat [Program Kepatuhan AWS](#).

Anda dapat mengunduh laporan audit pihak ketiga menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat [Mengunduh Laporan di AWS Artifak](#).

Tanggung jawab kepatuhan Anda saat menggunakan AWS layanan ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, dan hukum dan peraturan yang berlaku. AWS menyediakan sumber daya berikut untuk membantu kepatuhan:

- [Kepatuhan dan Tata Kelola Keamanan](#) – Panduan implementasi solusi ini membahas pertimbangan arsitektur serta memberikan langkah-langkah untuk menerapkan fitur keamanan dan kepatuhan.
- [Referensi Layanan yang Memenuhi Syarat HIPAA](#) — Daftar layanan yang memenuhi syarat HIPAA. Tidak semua AWS layanan memenuhi syarat HIPAA.
- [AWS Sumber Daya Kepatuhan](#) — Kumpulan buku kerja dan panduan ini mungkin berlaku untuk industri dan lokasi Anda.
- [AWS Panduan Kepatuhan Pelanggan](#) - Memahami model tanggung jawab bersama melalui lensa kepatuhan. Panduan ini merangkum praktik terbaik untuk mengamankan AWS layanan

dan memetakan panduan untuk kontrol keamanan di berbagai kerangka kerja (termasuk Institut Standar dan Teknologi Nasional (NIST), Dewan Standar Keamanan Industri Kartu Pembayaran (PCI), dan Organisasi Internasional untuk Standardisasi (ISO)).

- [Mengevaluasi Sumber Daya dengan Aturan](#) dalam Panduan Pengembang AWS Config — Layanan AWS Config menilai seberapa baik konfigurasi sumber daya Anda mematuhi praktik internal, pedoman industri, dan peraturan.
- [AWS Security Hub](#) — AWS Layanan ini memberikan pandangan komprehensif tentang keadaan keamanan Anda di dalamnya AWS. Security Hub menggunakan kontrol keamanan untuk mengevaluasi AWS sumber daya Anda dan untuk memeriksa kepatuhan Anda terhadap standar industri keamanan dan praktik terbaik. Untuk daftar layanan dan kontrol yang didukung, lihat [Referensi kontrol Security Hub](#).
- [Amazon GuardDuty](#) — AWS Layanan ini mendeteksi potensi ancaman terhadap AWS akun, beban kerja, kontainer, dan data Anda dengan memantau lingkungan Anda untuk aktivitas mencurigakan dan berbahaya. GuardDuty dapat membantu Anda mengatasi berbagai persyaratan kepatuhan, seperti PCI DSS, dengan memenuhi persyaratan deteksi intrusi yang diamanatkan oleh kerangka kerja kepatuhan tertentu.
- [AWS Audit Manager](#) — AWS Layanan ini membantu Anda terus mengaudit AWS penggunaan Anda untuk menyederhanakan cara Anda mengelola risiko dan kepatuhan terhadap peraturan dan standar industri.

Pertimbangan keamanan untuk Amazon Elastic Kubernetes Service

Berikut ini adalah pertimbangan untuk keamanan cloud, karena mempengaruhi Amazon EKS.

Topik

- [Keamanan infrastruktur di Amazon EKS](#)
- [Memahami ketahanan di kluster Amazon EKS](#)

Keamanan infrastruktur di Amazon EKS

Sebagai layanan terkelola, Amazon Elastic Kubernetes Service dilindungi oleh keamanan jaringan global. AWS Untuk informasi tentang layanan AWS keamanan dan cara AWS melindungi infrastruktur, lihat [Keamanan AWS Cloud](#). Untuk mendesain AWS lingkungan Anda menggunakan

praktik terbaik untuk keamanan infrastruktur, lihat [Perlindungan Infrastruktur dalam Kerangka Kerja](#) yang AWS Diarsiteksikan dengan Baik Pilar Keamanan.

Anda menggunakan panggilan API yang AWS dipublikasikan untuk mengakses Amazon EKS melalui jaringan. Klien harus mendukung hal-hal berikut:

- Keamanan Lapisan Pengangkutan (TLS). Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Sandi cocok dengan sistem kerahasiaan maju sempurna (perfect forward secrecy, PFS) seperti DHE (Ephemeral Diffie-Hellman) atau ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Sebagian besar sistem modern seperti Java 7 dan versi lebih baru mendukung mode-mode ini.

Selain itu, permintaan harus ditandatangani menggunakan ID kunci akses dan kunci akses rahasia yang terkait dengan prinsipal IAM. Atau Anda dapat menggunakan [AWS Security Token Service](#) (AWS STS) untuk menghasilkan kredensial keamanan sementara untuk menandatangani permintaan.

Ketika Anda membuat klaster Amazon EKS, Anda menentukan subnet VPC untuk digunakan oleh klaster Anda. Amazon EKS membutuhkan subnet, setidaknya di dua Availability Zone. Kami merekomendasikan VPC dengan subnet publik dan privat sehingga Kubernetes dapat membuat penyeimbang beban publik di subnet publik yang memuat lalu lintas keseimbangan ke Pod yang berjalan pada node yang berada di subnet pribadi.

Untuk informasi selengkapnya tentang pertimbangan VPC, lihat [the section called “Persyaratan VPC dan subnet”](#).

Jika Anda membuat grup VPC dan node dengan AWS CloudFormation templat yang disediakan dalam panduan [Memulai dengan Amazon EKS](#), maka grup keamanan bidang kontrol dan node Anda dikonfigurasi dengan pengaturan yang kami rekomendasikan.

Untuk informasi selengkapnya tentang pertimbangan grup keamanan, lihat [the section called “Persyaratan grup keamanan”](#).

Ketika Anda membuat klaster baru, Amazon EKS membuat titik akhir untuk server API Kubernetes terkelola yang Anda gunakan untuk berkomunikasi dengan klaster Anda (alat pengelolaan Kubernetes seperti `kubectl`). Secara default, endpoint server API ini bersifat publik ke internet, dan akses ke server API diamankan menggunakan kombinasi AWS Identity and Access Management (IAM) dan Access Management (IAM) dan native [Kubernetes](#) Role Based Access Control (RBAC).

Anda dapat mengaktifkan akses privat ke server API Kubernetes sehingga semua komunikasi antara simpul Anda dan server API tetap berada di dalam VPC Anda. Anda dapat membatasi alamat IP

yang dapat mengakses server API Anda dari internet, atau menonaktifkan sepenuhnya akses internet ke server API.

Untuk informasi selengkapnya tentang cara memodifikasi akses titik akhir klaster, lihat [the section called “Memodifikasi akses titik akhir klaster”](#).

[Anda dapat menerapkan kebijakan jaringan Kubernetes dengan Amazon VPC CNI atau alat pihak ketiga seperti Project Calico](#). Untuk informasi selengkapnya tentang menggunakan Amazon VPC CNI untuk kebijakan jaringan, lihat [the section called “Kebijakan Kubernetes”](#) Project Calico merupakan proyek sumber terbuka pihak ketiga. Untuk informasi selengkapnya, lihat [Project Calico documentation](#).

Akses Amazon EKS menggunakan AWS PrivateLink

Anda dapat menggunakannya AWS PrivateLink untuk membuat koneksi pribadi antara VPC Anda dan Amazon Elastic Kubernetes Service. Anda dapat mengakses Amazon EKS seolah-olah berada di VPC Anda, tanpa menggunakan gateway internet, perangkat NAT, koneksi VPN, atau koneksi Direct AWS Connect. Instans di VPC Anda tidak memerlukan alamat IP publik untuk mengakses Amazon EKS.

Anda membuat koneksi pribadi ini dengan membuat titik akhir antarmuka yang didukung oleh AWS PrivateLink. Kami membuat antarmuka jaringan endpoint di setiap subnet yang Anda aktifkan untuk titik akhir antarmuka. Ini adalah antarmuka jaringan yang dikelola pemohon yang berfungsi sebagai titik masuk untuk lalu lintas yang ditujukan untuk Amazon EKS.

Untuk informasi selengkapnya, lihat [Akses AWS layanan melalui AWS PrivateLink AWS](#) PrivateLink Panduan.

Sebelum Anda mulai

Sebelum memulai, pastikan Anda telah melakukan tugas-tugas berikut:

- Tinjau [Akses AWS layanan menggunakan titik akhir VPC antarmuka](#) di Panduan AWS PrivateLink

Pertimbangan

- Support and Limitations: Endpoint antarmuka Amazon EKS memungkinkan akses aman ke semua tindakan Amazon EKS API dari VPC Anda tetapi dilengkapi dengan batasan khusus: mereka tidak mendukung akses ke APIs Kubernetes, karena ini memiliki titik akhir pribadi yang terpisah, Anda tidak dapat mengonfigurasi Amazon EKS agar hanya dapat diakses melalui titik akhir antarmuka.

- **Harga:** Menggunakan titik akhir antarmuka untuk Amazon EKS menimbulkan AWS PrivateLink biaya standar: biaya per jam untuk setiap titik akhir yang disediakan di setiap Zona Ketersediaan, biaya pemrosesan data untuk lalu lintas melalui titik akhir. Untuk mempelajari lebih lanjut, lihat [AWS PrivateLink harga](#).
- **Keamanan dan Kontrol Akses:** Sebaiknya tingkatkan keamanan dan kontrol akses dengan konfigurasi tambahan ini—gunakan kebijakan titik akhir VPC untuk mengontrol akses ke Amazon EKS melalui titik akhir antarmuka, mengaitkan grup keamanan dengan antarmuka jaringan titik akhir untuk mengelola lalu lintas, menggunakan log aliran VPC untuk menangkap dan memantau lalu lintas IP ke dan dari titik akhir antarmuka, dengan log yang dapat dipublikasikan ke Amazon atau Amazon S3. CloudWatch Untuk mempelajari lebih lanjut, lihat [Mengontrol akses ke titik akhir VPC menggunakan kebijakan titik akhir dan Mencatat lalu lintas IP menggunakan Log Aliran VPC](#).
- **Opsi Konektivitas:** Titik akhir antarmuka menawarkan opsi konektivitas fleksibel menggunakan akses lokal (sambungkan pusat data lokal Anda ke VPC dengan titik akhir antarmuka menggunakan Direct AWS Connect atau AWS Site-to-Site VPN) atau melalui konektivitas antar-VPC (gunakan Transit AWS Gateway atau peering VPC untuk VPCs menghubungkan orang lain ke VPC dengan titik akhir antarmuka, menjaga lalu lintas dalam jaringan). AWS
- **Dukungan Versi IP:** Titik akhir yang dibuat sebelum Agustus 2024 mendukung hanya IPv4 menggunakan eks.region.amazonaws.com. Titik akhir baru yang dibuat setelah Agustus 2024 mendukung dual-stack IPv4 dan IPv6 (misalnya, eks.region.amazonaws.com, eks.region.api.aws).
- **Ketersediaan Regional:** AWS PrivateLink untuk EKS API tidak tersedia di wilayah Asia Pasifik (Malaysia) (ap-tenggara 5), Asia Pasifik (Thailand) (ap-tenggara - 7), Meksiko (Tengah) (mx-sentral-1), dan Asia Pasifik (Taipei) (ap-timur-2). AWS PrivateLink dukungan untuk eks-auth (EKS Pod Identity) tersedia di wilayah Asia Pasifik (Malaysia) (ap-tenggara 5).

Buat titik akhir antarmuka untuk Amazon EKS

Anda dapat membuat titik akhir antarmuka untuk Amazon EKS menggunakan konsol Amazon VPC atau Antarmuka Baris AWS Perintah (AWS CLI). Untuk informasi selengkapnya, lihat [Membuat titik akhir VPC di Panduan](#). AWS PrivateLink

Buat titik akhir antarmuka untuk Amazon EKS menggunakan nama layanan berikut:

EKS API

- com.amazonaws.region-code.eks
- com.amazonaws.region-code.eks-fips (untuk titik akhir yang sesuai dengan FIPS)

EKS Auth API (Identitas EKS Pod)

- `com.amazonaws.region-code.eks-auth`

Fitur DNS pribadi untuk titik akhir antarmuka Amazon EKS

Fitur DNS pribadi, diaktifkan secara default untuk titik akhir antarmuka Amazon EKS dan AWS layanan lainnya, memfasilitasi permintaan API yang aman dan pribadi menggunakan nama DNS Regional default. Fitur ini memastikan bahwa panggilan API dirutekan melalui titik akhir antarmuka melalui AWS jaringan pribadi, meningkatkan keamanan dan kinerja.

Fitur DNS pribadi aktif secara otomatis saat Anda membuat titik akhir antarmuka untuk Amazon EKS atau layanan lainnya. AWS Untuk mengaktifkan, Anda perlu mengkonfigurasi VPC Anda dengan benar dengan mengatur atribut tertentu:

- `enableDnsHostnames`: Memungkinkan instance dalam VPC memiliki nama host DNS.
- `enableDnsSupport`: Mengaktifkan resolusi DNS di seluruh VPC.

Untuk step-by-step petunjuk untuk memeriksa atau mengubah pengaturan ini, lihat [Melihat dan memperbarui atribut DNS untuk VPC Anda](#).

Nama DNS dan jenis alamat IP

Dengan mengaktifkan fitur DNS pribadi, Anda dapat menggunakan nama DNS tertentu untuk terhubung ke Amazon EKS, dan opsi ini berkembang seiring waktu:

- `eks.region.amazonaws.com`: Nama DNS tradisional, hanya diselesaikan ke alamat sebelum Agustus 2024. IPv4 Untuk titik akhir yang ada yang diperbarui ke dual-stack, nama ini diselesaikan ke keduanya dan alamat. IPv4 IPv6
- `eks.region.api.aws`: Tersedia untuk titik akhir baru yang dibuat setelah Agustus 2024, nama DNS tumpukan ganda ini diselesaikan ke keduanya dan alamat. IPv4 IPv6

Setelah Agustus 2024, titik akhir antarmuka baru hadir dengan dua nama DNS, dan Anda dapat memilih jenis alamat IP dual-stack. Untuk titik akhir yang ada, memperbarui ke dual-stack memodifikasi `eks.region.amazonaws.com` untuk mendukung keduanya dan. IPv4 IPv6

Menggunakan fitur DNS Pribadi

Setelah dikonfigurasi, fitur DNS pribadi dapat diintegrasikan ke dalam alur kerja Anda, menawarkan kemampuan berikut:

- **Permintaan API:** Gunakan nama DNS Regional default, baik `eks.region.amazonaws.com` atau `eks.region.api.aws`, berdasarkan penyiapan titik akhir Anda untuk membuat permintaan API ke Amazon EKS.
- **Kompatibilitas Aplikasi:** Aplikasi Anda yang ada yang memanggil EKS tidak APIs memerlukan perubahan untuk memanfaatkan fitur ini.
- **AWS CLI dengan Dual-Stack:** Untuk menggunakan titik akhir dual-stack dengan AWS CLI, lihat konfigurasi titik akhir [Dual-stack dan FIPS di](#) Panduan Referensi Alat dan . AWS SDKs
- **Perutean Otomatis:** Setiap panggilan ke titik akhir layanan default Amazon EKS secara otomatis diarahkan melalui titik akhir antarmuka, memastikan konektivitas pribadi dan aman.

Memahami ketahanan di kluster Amazon EKS

Infrastruktur AWS global dibangun di sekitar AWS Wilayah dan Zona Ketersediaan. AWS Wilayah menyediakan beberapa Availability Zone yang terpisah secara fisik dan terisolasi, yang terhubung dengan latensi rendah, throughput tinggi, dan jaringan yang sangat redundan. Dengan Zona Ketersediaan, Anda dapat merancang dan mengoperasikan aplikasi dan basis data yang secara otomatis melakukan failover di antara Zona Ketersediaan tanpa gangguan. Zona Ketersediaan memiliki ketersediaan dan toleransi kesalahan yang lebih baik, dan dapat diskalakan dibandingkan infrastruktur biasa yang terdiri dari satu atau beberapa pusat data.

Amazon EKS menjalankan dan menskalakan pesawat kontrol Kubernetes di beberapa AWS Availability Zone untuk memastikan ketersediaan yang tinggi. Amazon EKS secara otomatis menskalakan instans pesawat kontrol berdasarkan beban, mendeteksi, dan mengganti instans bidang kontrol yang tidak sehat, dan secara otomatis menambal bidang kontrol. Setelah Anda memulai pembaruan versi, Amazon EKS memperbarui bidang kontrol untuk Anda, menjaga ketersediaan tinggi bidang kontrol selama pembaruan.

Bidang kontrol ini terdiri dari setidaknya dua instance server API dan tiga etcd instance yang berjalan di tiga Availability Zone dalam suatu AWS Region. Amazon EKS:

- Secara aktif memantau beban pada instans bidang kendali dan secara otomatis menskalakannya guna memastikan kinerja yang tinggi.

- Secara otomatis mendeteksi dan mengganti instans bidang kontrol yang tidak sehat, memulai ulang di seluruh Availability Zone dalam Wilayah sesuai kebutuhan. AWS
- Memanfaatkan arsitektur AWS Daerah untuk menjaga ketersediaan tinggi. Karena itu, Amazon EKS mampu menawarkan [SLA untuk ketersediaan titik akhir server API](#).

Untuk informasi selengkapnya tentang AWS Wilayah dan Availability Zone, lihat [infrastruktur AWS global](#).

Pertimbangan keamanan untuk Kubernetes

Berikut ini adalah pertimbangan untuk keamanan di cloud, karena mereka mempengaruhi Kubernetes di cluster Amazon EKS. Untuk tinjauan mendalam tentang kontrol dan praktik keamanan di Kubernetes, lihat [Cloud Native Security dan Kubernetes di dokumentasi Kubernetes](#).

Topik

- [Amankan beban kerja dengan sertifikat Kubernetes](#)
- [Memahami peran dan pengguna RBAC yang dibuat Amazon EKS](#)
- [Enkripsi rahasia Kubernetes dengan KMS di cluster yang ada](#)
- [Gunakan AWS rahasia Secrets Manager dengan Amazon EKS Pods](#)
- [Enkripsi amplop default untuk semua Data API Kubernetes](#)

Amankan beban kerja dengan sertifikat Kubernetes

[Kubernetes Certificates API mengotomatiskan penyediaan kredensial X.509](#). API ini memiliki antarmuka baris perintah untuk klien API Kubernetes untuk meminta dan memperoleh sertifikat [X.509](#) dari [Certificate](#) Authority (CA). Anda dapat menggunakan sumber daya `CertificateSigningRequest` (CSR) untuk meminta penandatanganan yang dilambangkan menandatangani sertifikat. Permintaan Anda disetujui atau ditolak sebelum ditandatangani. Kubernetes mendukung penandatanganan bawaan dan penandatanganan khusus dengan perilaku yang terdefinisi dengan baik. Dengan cara ini, klien dapat memprediksi apa yang terjadi pada mereka CSRs. Untuk mempelajari selengkapnya tentang penandatanganan sertifikat, lihat [permintaan penandatanganan](#).

Salah satu penandatanganan bawaan adalah `kubernetes.io/legacy-unknown.v1beta1` API sumber daya CSR menghormati penandatanganan yang tidak dikenal lama ini. Namun, v1 API CSR yang stabil tidak memungkinkan `signatureName` untuk disetel ke `kubernetes.io/legacy-unknown`.

Jika ingin menggunakan Amazon EKS CA untuk membuat sertifikat di klaster, Anda harus menggunakan penandatanganan khusus. Untuk menggunakan versi CSR v1 API dan menghasilkan sertifikat baru, Anda harus memigrasikan manifes dan klien API yang ada. Sertifikat yang ada yang dibuat dengan v1beta1 API yang ada valid dan berfungsi hingga sertifikat kedaluwarsa. Ini termasuk yang berikut:

- Distribusi kepercayaan: Tidak ada. Tidak ada kepercayaan atau distribusi standar untuk penandatanganan ini di klaster Kubernetes.
- Subjek yang diizinkan: Apa saja
- Ekstensi x509 yang diizinkan: Menghormati subjectAltName dan ekstensi penggunaan kunci dan membuang ekstensi lainnya
- Penggunaan kunci yang diizinkan: Tidak boleh menyertakan penggunaan di luar ["encipherment kunci", "tanda tangan digital", "autentikasi server"]

 Note

Penandatanganan sertifikat klien tidak didukung.

- Kedaluwarsa/masa pakai sertifikat: 1 tahun (default dan maksimum)
- CA bit diizinkan/tidak diizinkan: Tidak diizinkan

Contoh pembuatan CSR dengan SignerName

Langkah-langkah ini menunjukkan cara menghasilkan sertifikat penyajian untuk `myserver.default.svc` menggunakan `signerName: beta.eks.amazonaws.com/app-serving` nama DNS. Gunakan ini sebagai panduan untuk lingkungan Anda sendiri.

1. Jalankan `openssl genrsa -out myserver.key 2048` perintah untuk menghasilkan kunci pribadi RSA.

```
openssl genrsa -out myserver.key 2048
```

2. Jalankan perintah berikut untuk menghasilkan permintaan sertifikat.

```
openssl req -new -key myserver.key -out myserver.csr -subj "/CN=myserver.default.svc"
```

3. Hasilkan base64 nilai untuk permintaan CSR dan simpan dalam variabel untuk digunakan di langkah selanjutnya.

```
base_64=$(cat myserver.csr | base64 -w 0 | tr -d "
")
```

4. Jalankan perintah berikut untuk membuat file bernama `mycsr.yaml`. Dalam contoh berikut, `beta.eks.amazonaws.com/app-serving` adalah `signerName`.

```
cat >mycsr.yaml <<EOF
apiVersion: certificates.k8s.io/v1
kind: CertificateSigningRequest
metadata:
  name: myserver
spec:
  request: $base_64
  signerName: beta.eks.amazonaws.com/app-serving
  usages:
    - digital signature
    - key encipherment
    - server auth
EOF
```

5. Kirim CSR.

```
kubectl apply -f mycsr.yaml
```

6. Menyetujui sertifikat penyajian.

```
kubectl certificate approve myserver
```

7. Verifikasi bahwa sertifikat telah dikeluarkan.

```
kubectl get csr myserver
```

Contoh output adalah sebagai berikut.

NAME	AGE	SIGNERNAME	REQUESTOR	CONDITION
myserver	3m20s	beta.eks.amazonaws.com/app-serving	kubernetes-admin	Approved, Issued

8. Ekspor sertifikat yang dikeluarkan.

```
kubectl get csr myserver -o jsonpath='{.status.certificate}' | base64 -d > myserver.crt
```

Memahami peran dan pengguna RBAC yang dibuat Amazon EKS

Saat Anda membuat klaster Kubernetes, beberapa identitas Kubernetes default dibuat di klaster tersebut agar Kubernetes berfungsi dengan baik. Amazon EKS membuat identitas Kubernetes untuk setiap komponen defaultnya. Identitas menyediakan kontrol otorisasi berbasis peran Kubernetes (RBAC) untuk komponen cluster. Untuk informasi selengkapnya, lihat [Menggunakan Otorisasi RBAC](#) dalam dokumentasi Kubernetes.

Saat Anda menginstal [add-on](#) opsional ke klaster Anda, identitas Kubernetes tambahan mungkin ditambahkan ke klaster Anda. Untuk informasi selengkapnya tentang identitas yang tidak dibahas oleh topik ini, lihat dokumentasi untuk add-on.

Anda dapat melihat daftar identitas Kubernetes yang dibuat Amazon EKS di klaster Anda menggunakan alat baris perintah AWS Management Console atau `kubectl`. Semua identitas pengguna muncul di log kube audit yang tersedia untuk Anda melalui Amazon CloudWatch.

AWS Management Console

Prasyarat

[Prinsipal IAM](#) yang Anda gunakan harus memiliki izin yang dijelaskan dalam Izin yang [diperlukan](#).

Untuk melihat identitas yang dibuat Amazon EKS menggunakan AWS Management Console

1. Buka [konsol Amazon EKS](#).
2. Dalam daftar Clusters, pilih cluster yang berisi identitas yang ingin Anda lihat.
3. Pilih tab Sumber Daya.
4. Di bawah Jenis sumber daya, pilih Otorisasi.
5. Pilih, ClusterRoles, ClusterRoleBindings, Peran, atau RoleBindings. Semua sumber daya yang diawali dengan eks dibuat oleh Amazon EKS. Sumber daya identitas tambahan yang dibuat Amazon EKS adalah:
 - ClusterRoleDan ClusterRoleBinding bernama aws-node. Sumber daya aws-node mendukung [plugin Amazon VPC CNI untuk Kubernetes](#), yang diinstal Amazon EKS di semua cluster.

- Yang `ClusterRole` bernama `vpc-resource-controller-role` dan `ClusterRoleBinding` bernama `vpc-resource-controller-rolebinding`. Sumber daya ini mendukung [pengontrol sumber daya VPC](#) Amazon, yang dipasang Amazon EKS di semua cluster.

Selain sumber daya yang Anda lihat di konsol, identitas pengguna khusus berikut ada di kluster Anda, meskipun tidak terlihat dalam konfigurasi cluster:

- **`eks:cluster-bootstrap`**- Digunakan untuk `kubectl` operasi selama bootstrap cluster.
 - **`eks:support-engineer`**— Digunakan untuk operasi manajemen cluster.
6. Pilih sumber daya tertentu untuk melihat detailnya. Secara default, Anda ditampilkan informasi dalam tampilan Terstruktur. Di sudut kanan atas halaman detail, Anda dapat memilih Tampilan mentah untuk melihat semua informasi sumber daya.

Kubectl

Prasyarat

Entitas yang Anda gunakan (AWS Identity and Access Management (IAM) and Access Management (IAM) atau OpenID Connect (OIDC)) untuk mencantumkan resource Kubernetes di kluster harus diautentikasi oleh IAM atau penyedia identitas OIDC Anda. Entitas harus diberikan izin untuk menggunakan Kubernetes `get` dan `list` kata kerja untuk `Role`, `ClusterRole`, `RoleBinding`, dan `ClusterRoleBinding` sumber daya di kluster Anda yang Anda inginkan untuk dikerjakan oleh entitas tersebut. Untuk informasi selengkapnya tentang pemberian entitas IAM akses ke kluster Anda, lihat. [the section called “Akses API Kubernetes”](#) Untuk informasi selengkapnya tentang pemberian entitas yang diautentikasi oleh penyedia OIDC Anda sendiri akses ke kluster Anda, lihat. [the section called “Tautkan penyedia OIDC”](#)

Untuk melihat identitas Amazon EKS yang dibuat menggunakan **`kubectl`**

Jalankan perintah untuk jenis sumber daya yang ingin Anda lihat. Semua sumber daya yang dikembalikan yang diawali dengan `eks` dibuat oleh Amazon EKS. Selain sumber daya yang dikembalikan dalam output dari perintah, identitas pengguna khusus berikut ada di cluster Anda, meskipun tidak terlihat dalam konfigurasi cluster:

- **`eks:cluster-bootstrap`**- Digunakan untuk `kubectl` operasi selama bootstrap cluster.
- **`eks:support-engineer`**— Digunakan untuk operasi manajemen cluster.

ClusterRoles— **ClusterRoles** dicakup ke klaster Anda, jadi izin apa pun yang diberikan untuk peran berlaku untuk sumber daya di namespace Kubernetes apa pun di klaster.

Perintah berikut mengembalikan semua Kubernetes Amazon EKS yang dibuat **ClusterRoles** di klaster Anda.

```
kubectl get clusterroles | grep eks
```

Selain **ClusterRoles** dikembalikan dalam output yang diawali dengan, berikut ini **ClusterRoles** ada.

- **aws-node**— Ini **ClusterRole** mendukung [plugin Amazon VPC CNI untuk Kubernetes](#), yang dipasang Amazon EKS di semua cluster.
- **vpc-resource-controller-role**— Ini **ClusterRole** mendukung [pengontrol sumber daya Amazon VPC](#), yang dipasang Amazon EKS di semua cluster.

Untuk melihat spesifikasi untuk a**ClusterRole**, ganti **eks:k8s-metrics** dalam perintah berikut dengan **ClusterRole** dikembalikan dalam output dari perintah sebelumnya. Contoh berikut mengembalikan spesifikasi untuk **eks:k8s-metricsClusterRole**.

```
kubectl describe clusterrole eks:k8s-metrics
```

Contoh output adalah sebagai berikut.

```
Name:          eks:k8s-metrics
Labels:        <none>
Annotations:   <none>
PolicyRule:
  Resources      Non-Resource URLs  Resource Names  Verbs
  -----
  endpoints      []                 []              [list]
  nodes          []                 []              [list]
  pods           []                 []              [list]
  deployments.apps []                 []              [list]
```

ClusterRoleBindings— **ClusterRoleBindings** dicakup ke cluster Anda.

Perintah berikut mengembalikan semua Kubernetes Amazon EKS yang dibuat **ClusterRoleBindings** di klaster Anda.


```
kubectl get clusterrolebindings | grep eks
```

Selain ClusterRoleBindings dikembalikan dalam output, berikut ini ClusterRoleBindings ada.

- **aws-node**— Ini ClusterRoleBinding mendukung [plugin Amazon VPC CNI untuk Kubernetes](#), yang dipasang Amazon EKS di semua cluster.
- **vpc-resource-controller-rolebinding**— Ini ClusterRoleBinding mendukung [pengontrol sumber daya Amazon VPC](#), yang dipasang Amazon EKS di semua cluster.

Untuk melihat spesifikasi untuk aClusterRoleBinding, ganti *eks:k8s-metrics* dalam perintah berikut dengan ClusterRoleBinding dikembalikan dalam output dari perintah sebelumnya. Contoh berikut mengembalikan spesifikasi untuk *eks:k8s-metrics* ClusterRoleBinding.

```
kubectl describe clusterrolebinding eks:k8s-metrics
```

Contoh output adalah sebagai berikut.

```
Name:          eks:k8s-metrics
Labels:        <none>
Annotations:   <none>
Role:
  Kind:  ClusterRole
  Name:  eks:k8s-metrics
Subjects:
  Kind  Name              Namespace
  ----  -
  User  eks:k8s-metrics
```

Peran — Roles dicakup ke namespace Kubernetes. Semua Amazon EKS Roles yang dibuat dicakup ke namespace. kube-system

Perintah berikut mengembalikan semua Kubernetes Amazon EKS yang dibuat Roles di klaster Anda.

```
kubectl get roles -n kube-system | grep eks
```

Untuk melihat spesifikasi untuk aRole, ganti *eks:k8s-metrics* dalam perintah berikut dengan nama yang Role dikembalikan dalam output dari perintah sebelumnya. Contoh berikut mengembalikan spesifikasi untuk *eks:k8s-metrics*Role.

```
kubectl describe role eks:k8s-metrics -n kube-system
```

Contoh output adalah sebagai berikut.

```
Name:          eks:k8s-metrics
Labels:        <none>
Annotations:   <none>
PolicyRule:
  Resources      Non-Resource URLs  Resource Names      Verbs
  -----
  daemonsets.apps  []                  [aws-node]          [get]
  deployments.apps  []                  [vpc-resource-controller] [get]
```

RoleBindings— RoleBindings dicakup ke namespace Kubernetes. Semua Amazon EKS RoleBindings yang dibuat dicakup ke namespace. kube-system

Perintah berikut mengembalikan semua Kubernetes Amazon EKS yang dibuat RoleBindings di klaster Anda.

```
kubectl get rolebindings -n kube-system | grep eks
```

Untuk melihat spesifikasi untuk aRoleBinding, ganti *eks:k8s-metrics* dalam perintah berikut dengan RoleBinding dikembalikan dalam output dari perintah sebelumnya. Contoh berikut mengembalikan spesifikasi untuk *eks:k8s-metrics*RoleBinding.

```
kubectl describe rolebinding eks:k8s-metrics -n kube-system
```

Contoh output adalah sebagai berikut.

```
Name:          eks:k8s-metrics
Labels:        <none>
Annotations:   <none>
Role:
  Kind:  Role
  Name:  eks:k8s-metrics
Subjects:
```

Kind	Name	Namespace
----	----	-----
User	eks:k8s-metrics	

Enkripsi rahasia Kubernetes dengan KMS di cluster yang ada

Important

Prosedur ini hanya berlaku untuk klaster EKS yang menjalankan Kubernetes versi 1.27 atau lebih rendah. Jika Anda menjalankan Kubernetes versi 1.28 atau lebih tinggi, rahasia Kubernetes Anda dilindungi dengan enkripsi amplop secara default. Untuk informasi selengkapnya, lihat [the section called “Enkripsi amplop default untuk semua Data API Kubernetes”](#).

Jika Anda mengaktifkan [enkripsi rahasia](#), rahasia Kubernetes dienkripsi menggunakan kunci AWS KMS yang Anda pilih. Kunci KMS harus memenuhi ketentuan berikut:

- Simetris
- Dapat mengenkripsi dan mendekripsi data
- Dibuat di AWS Wilayah yang sama dengan cluster
- Jika kunci KMS dibuat di akun yang berbeda, [kepala sekolah IAM](#) harus memiliki akses ke kunci KMS.

Untuk informasi selengkapnya, lihat [Mengizinkan kepala IAM di akun lain menggunakan kunci KMS](#) di Panduan Pengembang Layanan Manajemen [AWS Kunci](#).

Warning

Anda tidak dapat menonaktifkan enkripsi rahasia setelah mengaktifkannya. Tindakan ini tidak dapat diubah.

eksctl

Prosedur ini hanya berlaku untuk klaster EKS yang menjalankan Kubernetes versi 1.27 atau lebih rendah. Untuk informasi selengkapnya, lihat [the section called “Enkripsi amplop default untuk semua Data API Kubernetes”](#).

Anda dapat mengaktifkan enkripsi dengan dua cara:

- Tambahkan enkripsi ke kluster Anda dengan satu perintah.

Untuk mengenkripsi ulang rahasia Anda secara otomatis, jalankan perintah berikut.

```
eksctl utils enable-secrets-encryption \  
  --cluster my-cluster \  
  --key-arn arn:aws:kms:region-code:account:key/key
```

Untuk memilih keluar dari mengenkripsi ulang rahasia Anda secara otomatis, jalankan perintah berikut.

```
eksctl utils enable-secrets-encryption  
  --cluster my-cluster \  
  --key-arn arn:aws:kms:region-code:account:key/key \  
  --encrypt-existing-secrets=false
```

- Tambahkan enkripsi ke cluster Anda dengan `kms-cluster.yaml` file.

```
apiVersion: eksctl.io/v1alpha5  
kind: ClusterConfig  
  
metadata:  
  name: my-cluster  
  region: region-code  
  
secretsEncryption:  
  keyARN: arn:aws:kms:region-code:account:key/key
```

Agar rahasia Anda mengenkripsi ulang secara otomatis, jalankan perintah berikut.

```
eksctl utils enable-secrets-encryption -f kms-cluster.yaml
```

Untuk memilih keluar dari enkripsi ulang rahasia Anda secara otomatis, jalankan perintah berikut.

```
eksctl utils enable-secrets-encryption -f kms-cluster.yaml --encrypt-existing-  
secrets=false
```

AWS Management Console

- a. Prosedur ini hanya berlaku untuk kluster EKS yang menjalankan Kubernetes versi 1.27 atau lebih rendah. Untuk informasi selengkapnya, lihat [the section called “Enkripsi amplop default untuk semua Data API Kubernetes”](#).
- b. Buka [konsol Amazon EKS](#).
- c. Pilih cluster yang ingin Anda tambahkan enkripsi KMS.
- d. Pilih tab Ikhtisar (ini dipilih secara default).
- e. Gulir ke bawah ke bagian enkripsi Rahasia dan pilih Aktifkan.
- f. Pilih kunci dari daftar dropdown dan pilih tombol Enable. Jika tidak ada kunci yang tercantum, Anda harus membuatnya dulu. Untuk informasi selengkapnya, lihat [Membuat kunci](#)
- g. Pilih tombol Konfirmasi untuk menggunakan tombol yang dipilih.

AWS CLI

- a. Prosedur ini hanya berlaku untuk kluster EKS yang menjalankan Kubernetes versi 1.27 atau lebih rendah. Untuk informasi selengkapnya, lihat [the section called “Enkripsi amplop default untuk semua Data API Kubernetes”](#).
- b. Kaitkan konfigurasi [enkripsi rahasia](#) dengan cluster Anda menggunakan perintah AWS CLI berikut. Ganti *example values* dengan milik Anda sendiri.

```
aws eks associate-encryption-config \
  --cluster-name my-cluster \
  --encryption-config '[{"resources":["secrets"],"provider":
{"keyArn":"arn:aws:kms:region-code:account:key/key"}}]'
```

Contoh output adalah sebagai berikut.

```
{
  "update": {
    "id": "3141b835-8103-423a-8e68-12c2521ffa4d",
    "status": "InProgress",
    "type": "AssociateEncryptionConfig",
    "params": [
      {
        "type": "EncryptionConfig",
        "value": "[{\"resources\":[\"secrets\"],\"provider\":{\"keyArn\":
\\\"arn:aws:kms:region-code:account:key/key\\\"} }]"
      }
    ]
  }
}
```

```

    }
  ],
  "createdAt": 1613754188.734,
  "errors": []
}
}

```

- c. Anda dapat memantau status pembaruan enkripsi Anda dengan perintah berikut. Gunakan spesifik `cluster name` dan `update ID` yang dikembalikan pada output sebelumnya. Ketika `Successful` status ditampilkan, pembaruan selesai.

```

aws eks describe-update \
  --region region-code \
  --name my-cluster \
  --update-id 3141b835-8103-423a-8e68-12c2521ffa4d

```

Contoh output adalah sebagai berikut.

```

{
  "update": {
    "id": "3141b835-8103-423a-8e68-12c2521ffa4d",
    "status": "Successful",
    "type": "AssociateEncryptionConfig",
    "params": [
      {
        "type": "EncryptionConfig",
        "value": "[{\"resources\":[\"secrets\"],\"provider\":{\"keyArn\":\
\\arn:aws:kms:region-code:account:key/key\\\"}}]"
      }
    ],
    "createdAt": 1613754188.734>,
    "errors": []
  }
}

```

- d. Untuk memverifikasi bahwa enkripsi diaktifkan di klaster Anda, jalankan perintah `describe-cluster`. Responsnya berisi `EncryptionConfig` string.

```

aws eks describe-cluster --region region-code --name my-cluster

```

Setelah mengaktifkan enkripsi di klaster, Anda harus mengenkripsi semua rahasia yang ada dengan kunci baru:

 Note

Jika Anda menggunakan `kubectl`, menjalankan perintah berikut hanya diperlukan jika Anda memilih untuk tidak mengenkripsi ulang rahasia Anda secara otomatis.

```
kubectl get secrets --all-namespaces -o json | kubectl annotate --overwrite -f - kms-encryption-timestamp="time value"
```

 Warning

Jika Anda mengaktifkan [enkripsi rahasia](#) untuk cluster yang ada dan kunci KMS yang Anda gunakan pernah dihapus, maka tidak ada cara untuk memulihkan cluster. Jika Anda menghapus kunci KMS, Anda secara permanen menempatkan cluster dalam keadaan terdegradasi. Untuk informasi selengkapnya, lihat [Menghapus kunci AWS KMS](#).

 Note

Secara default, `create-key` perintah membuat [kunci KMS enkripsi simetris](#) dengan kebijakan kunci yang memberikan akses admin root akun pada tindakan dan sumber daya AWS KMS. Jika Anda ingin mengurangi izin, pastikan bahwa `kms:CreateGrant` tindakan `kms:DescribeKey` dan diizinkan pada kebijakan untuk prinsipal yang memanggil `create-cluster` API.

Untuk cluster yang menggunakan Enkripsi Amplop KMS, `kms:CreateGrant` izin diperlukan. Kondisi `kms:GrantIsForAWSResource` ini tidak didukung untuk `CreateCluster` tindakan, dan tidak boleh digunakan dalam kebijakan KMS untuk mengontrol `kms:CreateGrant` izin bagi pengguna yang melakukan `CreateCluster`.

Gunakan AWS rahasia Secrets Manager dengan Amazon EKS Pods

Untuk menampilkan rahasia dari Secrets Manager dan parameter dari Parameter Store sebagai file yang dipasang di Amazon EKS Pods, Anda dapat menggunakan AWS Secrets and Configuration Provider (ASCP) untuk [Kubernetes Secrets](#) Store CSI Driver.

Dengan ASCP, Anda dapat menyimpan dan mengelola rahasia Anda di Secrets Manager dan kemudian mengambilnya melalui beban kerja Anda yang berjalan di Amazon EKS. Anda dapat menggunakan peran dan kebijakan IAM untuk membatasi akses ke rahasia Anda ke Pod Kubernetes tertentu dalam sebuah kluster. ASCP mengambil identitas Pod dan menukar identitas untuk peran IAM. ASCP mengasumsikan peran IAM dari Pod, dan kemudian dapat mengambil rahasia dari Secrets Manager yang diberi wewenang untuk peran tersebut.

Jika Anda menggunakan rotasi otomatis Secrets Manager untuk rahasia Anda, Anda juga dapat menggunakan fitur reconciler rotasi Driver Secrets Store CSI untuk memastikan Anda mengambil rahasia terbaru dari Secrets Manager.

Note

AWS Grup simpul Fargate (Fargate) tidak didukung.

Untuk informasi selengkapnya, lihat [Menggunakan rahasia Secrets Manager di Amazon EKS](#) di Panduan Pengguna AWS Secrets Manager.

Enkripsi amplop default untuk semua Data API Kubernetes

Amazon Elastic Kubernetes Service (Amazon EKS) menyediakan enkripsi amplop default untuk semua data API Kubernetes di kluster EKS yang menjalankan Kubernetes versi 1.28 atau lebih tinggi.

Enkripsi amplop melindungi data yang Anda simpan dengan server API Kubernetes. Misalnya, enkripsi amplop berlaku untuk konfigurasi kluster Kubernetes Anda, seperti. ConfigMaps Enkripsi amplop tidak berlaku untuk data pada node atau volume EBS. EKS sebelumnya mendukung enkripsi rahasia Kubernetes, dan sekarang enkripsi amplop ini meluas ke semua data API Kubernetes.

Ini memberikan pengalaman default terkelola yang mengimplementasikan defense-in-depth aplikasi Kubernetes Anda dan tidak memerlukan tindakan apa pun dari pihak Anda.

Amazon EKS menggunakan AWS [Key Management Service \(KMS\)](#) dengan [penyedia Kubernetes KMS v2](#) untuk lapisan keamanan tambahan ini dengan [kunci milik Amazon Web Services, dan opsi bagi Anda untuk membawa kunci terkelola pelanggan \(CMK\) Anda sendiri dari KMS](#). AWS

Memahami enkripsi amplop

Enkripsi amplop adalah proses mengenkripsi data teks biasa dengan kunci enkripsi data (DEK) sebelum dikirim ke datastore (etcd), dan kemudian mengenkripsi DEK dengan kunci KMS root yang disimpan dalam sistem KMS jarak jauh yang dikelola secara terpusat (KMS).AWS Ini adalah defense-in-depth strategi karena melindungi data dengan kunci enkripsi (DEK), dan kemudian menambahkan lapisan keamanan lain dengan melindungi DEK itu dengan kunci enkripsi terpisah yang disimpan dengan aman yang disebut kunci enkripsi kunci (KEK).

Bagaimana Amazon EKS mengaktifkan enkripsi amplop default dengan KMS v2 dan KMS AWS

[Amazon EKS menggunakan KMS v2 untuk mengimplementasikan enkripsi amplop default untuk semua data API di bidang kontrol Kubernetes yang dikelola sebelum disimpan dalam database etcd.](#)

Saat startup, server API cluster menghasilkan kunci enkripsi data (DEK) dari seed rahasia yang dikombinasikan dengan data yang dihasilkan secara acak. Juga saat startup, server API membuat panggilan ke plugin KMS untuk mengenkripsi benih DEK menggunakan kunci enkripsi kunci jarak jauh (KEK) dari AWS KMS. Ini adalah panggilan satu kali yang dijalankan saat startup server API dan pada rotasi KEK. Server API kemudian menyimpan benih DEK terenkripsi. Setelah ini, server API menggunakan benih DEK yang di-cache untuk menghasilkan penggunaan tunggal lainnya DEKs berdasarkan Fungsi Derivasi Kunci (KDF). Masing-masing yang dihasilkan DEKs ini kemudian digunakan hanya sekali untuk mengenkripsi satu sumber daya Kubernetes sebelum disimpan dalam etcd. Dengan penggunaan seed DEK yang di-cache terenkripsi di KMS v2, proses mengenkripsi sumber daya Kubernetes di server API menjadi lebih berkinerja dan hemat biaya.

Secara default, KEK ini dimiliki oleh AWS, tetapi Anda dapat secara opsional membawa milik Anda sendiri dari AWS KMS.

Diagram di bawah ini menggambarkan pembuatan dan enkripsi DEK saat startup server API.

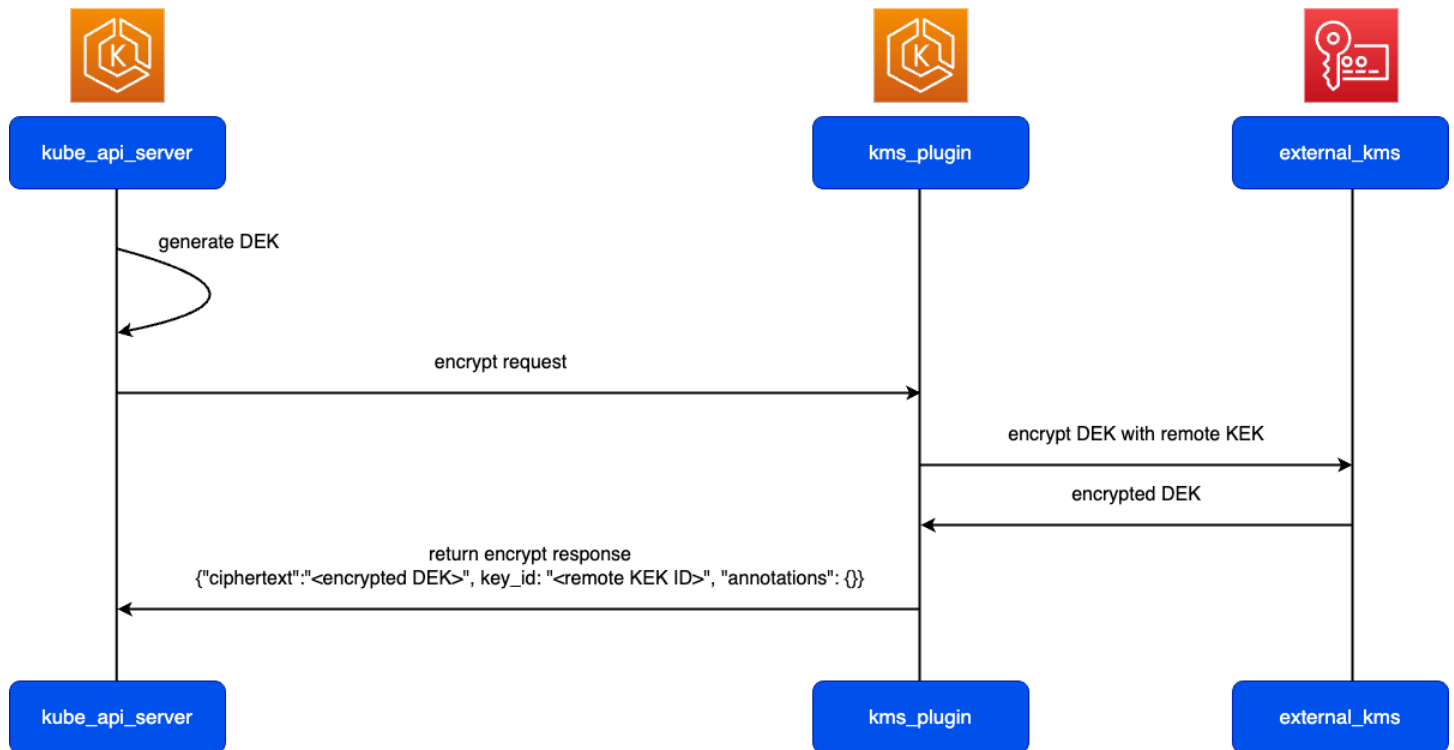
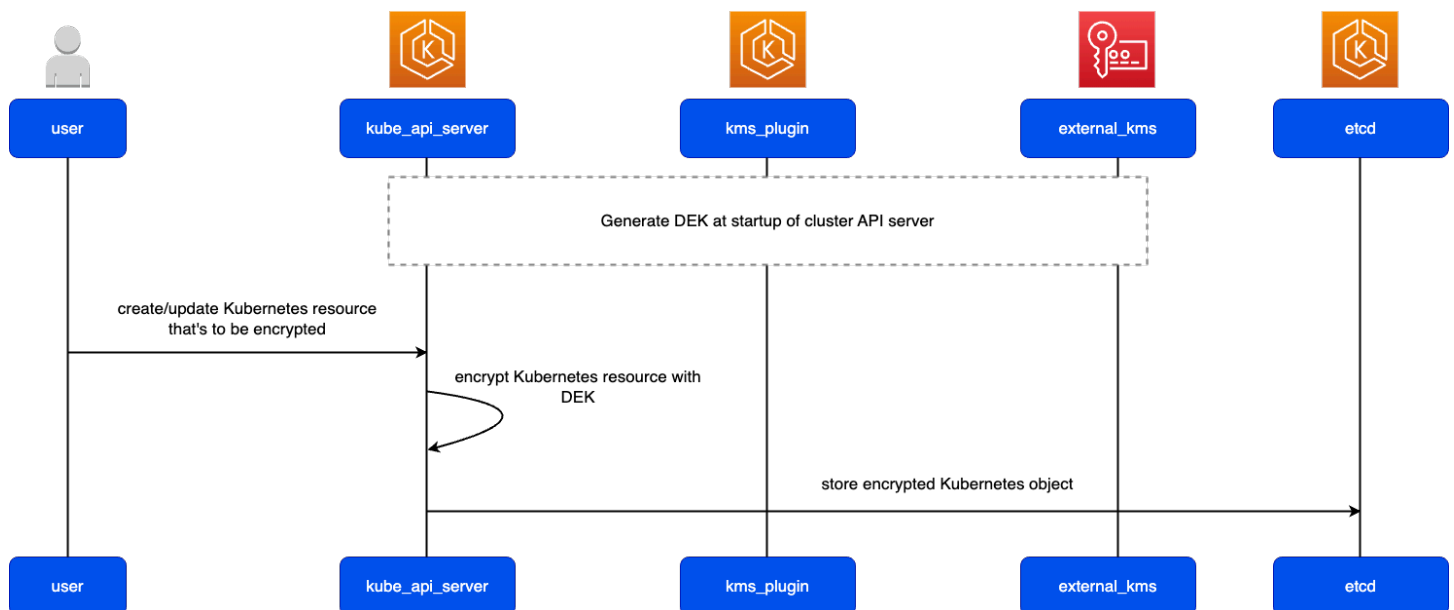


Diagram tingkat tinggi di bawah ini menggambarkan enkripsi sumber daya Kubernetes sebelum disimpan dalam etcd.



Pertanyaan umum

Bagaimana enkripsi amplop default meningkatkan postur keamanan cluster EKS saya?

Fitur ini mengurangi luas permukaan dan periode waktu di mana metadata dan konten pelanggan tidak dienkripsi. Dengan enkripsi amplop default, metadata dan konten pelanggan hanya pernah dalam keadaan tidak terenkripsi sementara di memori kube-apiserver sebelum disimpan dalam etcd. [Memori kube-apiserver diamankan melalui sistem Nitro](#). Amazon EKS hanya menggunakan [EC2 instans berbasis Nitro untuk bidang](#) kontrol Kubernetes yang dikelola. Instans ini memiliki desain kontrol keamanan yang mencegah sistem atau orang mengakses memori mereka.

Versi Kubernetes mana yang perlu saya jalankan untuk memiliki fitur ini?

Agar enkripsi amplop default diaktifkan, kluster Amazon EKS Anda harus menjalankan Kubernetes versi 1.28 atau yang lebih baru.

Apakah data saya masih aman jika saya menjalankan versi cluster Kubernetes yang tidak mendukung fitur ini?

Ya. Pada AWS, [keamanan adalah prioritas utama kami](#). Kami mendasarkan semua transformasi dan inovasi digital kami pada praktik operasional keamanan tertinggi, dan tetap berkomitmen untuk meningkatkan standar itu.

Semua data yang disimpan dalam etcd dienkripsi pada tingkat disk untuk setiap cluster EKS, terlepas dari versi Kubernetes yang sedang dijalankan. EKS menggunakan kunci root yang menghasilkan kunci enkripsi volume yang dikelola oleh layanan EKS. Selain itu, setiap kluster Amazon EKS dijalankan dalam VPC terisolasi menggunakan mesin virtual khusus cluster. Karena arsitektur ini, dan praktik kami seputar keamanan operasional, Amazon EKS telah [mencapai beberapa peringkat dan standar kepatuhan](#) termasuk kelayakan SOC 1,2,3, PCI-DSS, ISO, dan HIPAA. Peringkat dan standar kepatuhan ini dipertahankan untuk semua kluster EKS dengan atau tanpa enkripsi amplop default.

Bagaimana cara kerja enkripsi amplop di Amazon EKS?

Saat startup, server API cluster menghasilkan kunci enkripsi data (DEK) dari seed rahasia yang dikombinasikan dengan data yang dihasilkan secara acak. Juga saat startup, server API membuat panggilan ke plugin KMS untuk mengenkripsi DEK menggunakan kunci enkripsi kunci jarak jauh (KEK) dari AWS KMS. Ini adalah panggilan satu kali yang dijalankan saat startup server API dan pada rotasi KEK. Server API kemudian menyimpan benih DEK terenkripsi. Setelah ini, server API

menggunakan benih DEK yang di-cache untuk menghasilkan penggunaan tunggal lainnya DEKs berdasarkan Fungsi Derivasi Kunci (KDF). Masing-masing yang dihasilkan DEKs ini kemudian digunakan hanya sekali untuk mengenkripsi satu sumber daya Kubernetes sebelum disimpan dalam etcd.

Penting untuk dicatat bahwa ada panggilan tambahan yang dilakukan dari server API untuk memverifikasi kesehatan dan fungsionalitas normal integrasi AWS KMS. Pemeriksaan kesehatan tambahan ini terlihat di Anda AWS CloudTrail.

Apakah saya harus melakukan sesuatu atau mengubah izin apa pun agar fitur ini berfungsi di cluster EKS saya?

Tidak, Anda tidak perlu mengambil tindakan apa pun. Enkripsi amplop di Amazon EKS sekarang merupakan konfigurasi default yang diaktifkan di semua cluster yang menjalankan Kubernetes versi 1.28 atau lebih tinggi. Integrasi AWS KMS dibuat oleh server API Kubernetes yang dikelola oleh AWS. Ini berarti Anda tidak perlu mengonfigurasi izin apa pun untuk mulai menggunakan enkripsi KMS untuk cluster Anda.

Bagaimana saya bisa tahu jika enkripsi amplop default diaktifkan di cluster saya?

Jika Anda bermigrasi untuk menggunakan CMK Anda sendiri, maka Anda akan melihat ARN dari kunci KMS yang terkait dengan cluster Anda. Selain itu, Anda dapat melihat log AWS CloudTrail peristiwa yang terkait dengan penggunaan CMK cluster Anda.

Jika cluster Anda menggunakan kunci yang AWS dimiliki, maka ini akan dirinci di konsol EKS (tidak termasuk ARN kunci).

Bisakah AWS mengakses kunci yang AWS dimiliki yang digunakan untuk enkripsi amplop default di Amazon EKS?

Tidak. AWS memiliki kontrol keamanan yang ketat di Amazon EKS yang mencegah siapa pun mengakses kunci enkripsi teks biasa yang digunakan untuk mengamankan data dalam database etcd. Langkah-langkah keamanan ini juga diterapkan pada kunci KMS yang AWS dimiliki.

Apakah enkripsi amplop default diaktifkan di cluster EKS saya yang ada?

Jika Anda menjalankan kluster Amazon EKS dengan Kubernetes versi 1.28 atau lebih tinggi, enkripsi amplop semua data API Kubernetes diaktifkan. Untuk cluster yang ada, Amazon EKS menggunakan `eks:kms-storage-migrator` RBAC ClusterRole untuk memigrasikan data yang sebelumnya tidak dienkripsi dalam etcd ke status enkripsi baru ini.

Apa artinya ini jika saya sudah mengaktifkan enkripsi amplop untuk Rahasia di cluster EKS saya?

Jika Anda memiliki Customer Managed Key (CMK) di KMS yang digunakan untuk menyelubungi enkripsi Kubernetes Secrets Anda, kunci yang sama akan digunakan sebagai KEK untuk enkripsi amplop dari semua tipe data Kubernetes API di klaster Anda.

Apakah ada biaya tambahan untuk menjalankan cluster EKS dengan enkripsi amplop default?

Tidak ada biaya tambahan yang terkait dengan control plane Kubernetes yang dikelola jika Anda menggunakan [kunci milik Amazon Web Services](#) untuk enkripsi amplop default. Secara default, setiap kluster EKS yang menjalankan Kubernetes versi 1.28 atau yang lebih baru menggunakan kunci milik [Amazon Web Service](#). Namun, jika Anda menggunakan kunci AWS KMS Anda sendiri, [harga KMS normal akan berlaku](#).

Berapa biaya untuk menggunakan kunci AWS KMS saya sendiri untuk mengenkripsi data API Kubernetes di cluster saya?

Anda membayar \$1 per bulan untuk menyimpan kunci khusus apa pun yang Anda buat atau impor ke KMS. Biaya KMS untuk permintaan enkripsi dan dekripsi. Ada tingkat gratis 20.000 permintaan per bulan per akun dan Anda membayar \$0,03 per 10.000 permintaan di atas tingkat gratis per bulan. Ini berlaku di semua penggunaan KMS untuk akun, sehingga biaya penggunaan kunci AWS KMS Anda sendiri di klaster Anda akan dipengaruhi oleh penggunaan kunci ini pada cluster atau AWS sumber daya lain dalam akun Anda.

Apakah biaya KMS saya akan lebih tinggi sekarang karena kunci terkelola pelanggan (CMK) saya digunakan untuk membungkus enkripsi semua data API Kubernetes dan bukan hanya Rahasia?

Tidak. Implementasi kami dengan KMS v2 secara signifikan mengurangi jumlah panggilan yang dilakukan ke AWS KMS. Ini pada gilirannya akan mengurangi biaya yang terkait dengan CMK Anda terlepas dari data Kubernetes tambahan yang dienkripsi atau didekripsi di kluster EKS Anda.

Seperti yang dijelaskan di atas, seed DEK yang dihasilkan yang digunakan untuk enkripsi sumber daya Kubernetes disimpan secara lokal di cache server API Kubernetes setelah dienkripsi dengan KEK jarak jauh. Jika benih DEK terenkripsi tidak ada dalam cache server API, server API akan memanggil AWS KMS untuk mengenkripsi benih DEK. Server API kemudian menyimpan benih DEK terenkripsi untuk penggunaan masa depan di cluster tanpa memanggil KMS. Demikian pula, untuk permintaan dekripsi, server API akan memanggil AWS KMS untuk permintaan dekripsi pertama, setelah itu benih DEK yang didekripsi akan di-cache dan digunakan untuk operasi dekripsi masa depan.

Untuk informasi selengkapnya, lihat [KEP-3299: KMS v2 Improvements in the Kubernetes Enhancements](#) on. GitHub

Bisakah saya menggunakan kunci CMK yang sama untuk beberapa cluster Amazon EKS?

Ya. Untuk menggunakan kunci lagi, Anda dapat menautkannya ke cluster di wilayah yang sama dengan mengaitkan ARN dengan cluster selama pembuatan. Namun, jika Anda menggunakan CMK yang sama untuk beberapa kluster EKS, Anda harus menerapkan langkah-langkah yang diperlukan untuk mencegah penonaktifan CMK secara sewenang-wenang. Jika tidak, CMK yang dinonaktifkan yang terkait dengan beberapa kluster EKS akan memiliki cakupan dampak yang lebih luas pada cluster tergantung pada kunci itu.

Apa yang terjadi pada cluster EKS saya jika CMK saya menjadi tidak tersedia setelah enkripsi amplop default diaktifkan?

Jika Anda menonaktifkan kunci KMS, itu tidak dapat digunakan dalam operasi [kriptografi](#) apa pun. Tanpa akses ke CMK yang ada, server API tidak akan dapat mengenkripsi dan mempertahankan objek Kubernetes yang baru dibuat, serta mendekripsi objek Kubernetes yang sebelumnya dienkrpsi yang disimpan dalam etcd. Jika CMK dinonaktifkan, klaster akan segera ditempatkan dalam keadaan tidak sehat/terdegradasi di mana kami tidak akan dapat memenuhi [Komitmen Layanan](#) kami sampai Anda mengaktifkan kembali CMK terkait.

Ketika CMK dinonaktifkan, Anda akan menerima pemberitahuan tentang kesehatan klaster EKS Anda yang menurun dan kebutuhan untuk mengaktifkan kembali CMK Anda dalam waktu 30 hari setelah menonaktifkannya untuk memastikan pemulihan sumber daya pesawat kontrol Kubernetes Anda berhasil.

Bagaimana cara melindungi kluster EKS saya dari dampak CMK yang dinonaktifkan/dihapus?

Untuk melindungi klaster EKS Anda dari kejadian seperti itu, administrator kunci Anda harus mengelola akses ke operasi kunci KMS menggunakan kebijakan IAM dengan prinsip hak istimewa paling sedikit untuk mengurangi risiko penonaktifan sewenang-wenang atau penghapusan kunci yang terkait dengan kluster EKS. Selain itu, Anda dapat mengatur [CloudWatch alarm](#) untuk diberitahu tentang status CMK Anda.

Apakah cluster EKS saya akan dipulihkan jika saya mengaktifkan kembali CMK?

Untuk memastikan pemulihan klaster EKS Anda berhasil, kami sangat menyarankan untuk mengaktifkan kembali CMK Anda dalam 30 hari pertama setelah CMK dinonaktifkan. Namun,

pemulihan kluster EKS Anda yang berhasil juga akan bergantung pada apakah kluster mengalami perubahan yang melanggar API atau tidak karena peningkatan Kubernetes otomatis yang mungkin terjadi saat kluster dalam keadaan tidak sehat/terdegradasi.

Mengapa cluster EKS saya ditempatkan dalam keadaan tidak sehat/terdegradasi setelah menonaktifkan CMK?

Server API bidang kontrol EKS menggunakan kunci DEK yang dienkripsi dan di-cache dalam memori server API untuk mengenkripsi semua objek selama operasi pembuatan/pembaruan sebelum disimpan dalam etcd. Ketika objek yang ada diambil dari etcd, server API menggunakan kunci DEK cache yang sama dan mendekripsi objek sumber daya Kubernetes. Jika Anda menonaktifkan CMK, server API tidak akan melihat dampak langsung karena kunci DEK yang di-cache di memori server API. Namun, ketika instance server API dimulai ulang, itu tidak akan memiliki DEK yang di-cache dan perlu memanggil AWS KMS untuk mengenkripsi dan mendekripsi operasi. Tanpa CMK, proses ini akan gagal dengan kode kesalahan KMS_KEY_DISABLED, mencegah server API berhasil melakukan booting.

Apa yang terjadi pada cluster EKS saya jika saya menghapus CMK saya?

Menghapus kunci CMK yang terkait dengan kluster EKS Anda akan menurunkan kesehatannya setelah pemulihan. Tanpa CMK kluster Anda, server API tidak akan lagi dapat mengenkripsi dan mempertahankan objek Kubernetes baru, serta mendekripsi objek Kubernetes yang sebelumnya dienkripsi yang disimpan dalam database etcd. Anda hanya harus melanjutkan dengan menghapus kunci CMK untuk cluster EKS Anda ketika Anda yakin bahwa Anda tidak perlu menggunakan cluster EKS lagi.

Harap dicatat bahwa jika CMK tidak ditemukan (KMS_KEY_NOT_FOUND) atau hibah untuk CMK yang terkait dengan cluster Anda dicabut (KMS_GRANT_REVOKED), kluster Anda tidak akan dapat dipulihkan. Untuk informasi selengkapnya tentang kesehatan kluster dan kode kesalahan, lihat [Kesehatan kluster FAQs dan kode kesalahan dengan jalur resolusi](#).

Apakah saya masih akan dikenakan biaya untuk kluster EKS yang terdegradasi/tidak sehat karena saya menonaktifkan atau menghapus CMK saya?

Ya. Meskipun bidang kontrol EKS tidak akan dapat digunakan jika CMK dinonaktifkan, masih AWS akan menjalankan sumber daya infrastruktur khusus yang dialokasikan ke kluster EKS sampai dihapus oleh pelanggan. Selain itu, [Komitmen Layanan](#) kami tidak akan berlaku dalam keadaan seperti itu karena akan menjadi tindakan sukarela atau tidak bertindak oleh pelanggan yang mencegah kesehatan normal dan pengoperasian kluster EKS Anda.

Dapatkah cluster EKS saya ditingkatkan secara otomatis ketika dalam keadaan tidak sehat/terdegradasi karena CMK yang dinonaktifkan?

Ya. Namun, jika cluster Anda memiliki CMK yang dinonaktifkan, Anda akan memiliki periode 30 hari untuk mengaktifkannya kembali. Dalam periode 30 hari ini, klaster Kubernetes Anda tidak akan ditingkatkan secara otomatis. Namun, jika periode ini berlalu dan Anda belum mengaktifkan kembali CMK, klaster akan secara otomatis ditingkatkan ke versi berikutnya (n+1) yang berada dalam dukungan standar, mengikuti siklus hidup versi Kubernetes di EKS.

Kami sangat menyarankan untuk mengaktifkan kembali CMK yang dinonaktifkan dengan cepat ketika Anda mengetahui klaster yang terkena dampak. Penting untuk dicatat, bahwa meskipun EKS akan secara otomatis memutakhirkan klaster yang terkena dampak ini, tidak ada jaminan bahwa mereka akan berhasil pulih, terutama jika cluster mengalami beberapa peningkatan otomatis karena ini mungkin termasuk perubahan pada Kubernetes API dan perilaku tak terduga dalam proses bootstrap server API.

Bisakah saya menggunakan alias kunci KMS?

Ya. Amazon EKS [mendukung penggunaan alias kunci KMS](#). Alias adalah nama yang ramah untuk kunci [Amazon Web Service KMS](#). Misalnya, alias memungkinkan Anda merujuk ke kunci KMS sebagai kunci saya, bukan. **1234abcd-12ab-34cd-56ef-1234567890ab**

Apakah saya masih dapat mencadangkan dan memulihkan sumber daya klaster saya menggunakan solusi cadangan Kubernetes saya sendiri?

Ya. Anda dapat menggunakan solusi cadangan Kubernetes (seperti [Velero](#)) untuk pemulihan bencana klaster Kubernetes, migrasi data, dan perlindungan data. Jika Anda menjalankan solusi cadangan Kubernetes yang mengakses sumber daya cluster melalui server API, data apa pun yang diambil aplikasi akan didekripsi sebelum mencapai klien. Ini akan memungkinkan Anda untuk memulihkan sumber daya cluster di cluster Kubernetes lain.

Pertimbangan keamanan untuk Amazon EKS Auto Mode

Topik ini menjelaskan arsitektur keamanan, kontrol, dan praktik terbaik untuk Mode Otomatis Amazon EKS. Ketika organisasi menerapkan aplikasi kontainer dalam skala besar, mempertahankan postur keamanan yang kuat menjadi semakin kompleks. Mode Otomatis EKS mengimplementasikan kontrol keamanan otomatis dan terintegrasi dengan layanan AWS keamanan untuk membantu Anda melindungi infrastruktur klaster, beban kerja, dan data Anda. Melalui fitur keamanan bawaan seperti

manajemen siklus hidup node yang diberlakukan dan penerapan patch otomatis, Mode Otomatis EKS membantu Anda mempertahankan praktik terbaik keamanan sekaligus mengurangi overhead operasional.

Sebelum melanjutkan dengan topik ini, pastikan Anda terbiasa dengan konsep Mode Otomatis EKS dasar dan telah meninjau prasyarat untuk mengaktifkan Mode Otomatis EKS di cluster Anda. Untuk informasi umum tentang keamanan Amazon EKS, lihat [Keamanan](#).

Amazon EKS Auto Mode dibangun di atas fondasi keamanan Amazon EKS yang ada sambil memperkenalkan kontrol keamanan otomatis tambahan untuk instans EC2 terkelola.

Keamanan dan otentikasi API

Amazon EKS Auto Mode menggunakan mekanisme keamanan AWS platform untuk mengamankan dan mengautentikasi panggilan ke Amazon EKS API.

- Akses ke API Kubernetes diamankan melalui entri akses EKS, yang terintegrasi dengan identitas IAM. AWS
 - Untuk informasi selengkapnya, lihat [the section called “Berikan izin”](#).
- Pelanggan dapat menerapkan kontrol akses berbutir halus ke titik akhir API Kubernetes melalui konfigurasi entri akses EKS.

Keamanan jaringan

Amazon EKS Auto Mode mendukung beberapa lapisan keamanan jaringan:

- Integrasi VPC
 - Beroperasi dalam Amazon Virtual Private Cloud (VPC)
 - Mendukung konfigurasi VPC kustom dan tata letak subnet
 - Mengaktifkan jaringan pribadi antara komponen cluster
 - Untuk informasi selengkapnya, lihat [Mengelola tanggung jawab keamanan untuk Amazon Virtual Private Cloud](#)
- Kebijakan Jaringan
 - Dukungan asli untuk Kebijakan Jaringan Kubernetes
 - Kemampuan untuk menentukan aturan lalu lintas jaringan granular
 - Untuk informasi selengkapnya, silakan lihat [the section called “Kebijakan Kubernetes”](#)

EC2 keamanan instance terkelola

Amazon EKS Auto Mode mengoperasikan instans EC2 terkelola dengan kontrol keamanan berikut:

EC2 keamanan

- EC2 instans terkelola mempertahankan fitur keamanan Amazon EC2.
- Untuk informasi selengkapnya tentang instans EC2 terkelola, lihat [Keamanan di Amazon EC2](#).

Manajemen siklus hidup instans

EC2 instans terkelola yang dioperasikan oleh Mode Otomatis EKS memiliki masa pakai maksimum 21 hari. Mode Otomatis Amazon EKS secara otomatis menghentikan instans yang melebihi masa pakai ini. Batas siklus hidup ini membantu mencegah penyimpangan konfigurasi dan mempertahankan postur keamanan.

Perlindungan data

- Amazon EC2 Instance Storage dienkripsi, ini adalah penyimpanan yang langsung dilampirkan ke instance. Untuk informasi selengkapnya, lihat [Perlindungan data di Amazon EC2](#).
- Mode Otomatis EKS mengelola volume yang dilampirkan ke EC2 instance pada waktu pembuatan, termasuk volume root dan data. Mode Otomatis EKS tidak sepenuhnya mengelola volume EBS yang dibuat menggunakan fitur penyimpanan persisten Kubernetes.

Manajemen tambalan

- Mode Otomatis Amazon EKS secara otomatis menerapkan tambalan ke instans terkelola.
- Patch meliputi:
 - Pembaruan sistem operasi
 - Patch keamanan
 - Komponen Mode Otomatis Amazon EKS

 Note

Pelanggan bertanggung jawab untuk mengamankan dan memperbarui beban kerja yang berjalan pada instans ini.

Kontrol akses

- Akses instans langsung dibatasi:
 - Akses SSH tidak tersedia.
 - AWS Akses Systems Manager Session Manager (SSM) tidak tersedia.
- Operasi manajemen dilakukan melalui Amazon EKS API dan Kubernetes API.

Manajemen sumber daya otomatis

Amazon EKS Auto Mode tidak sepenuhnya mengelola Volume Amazon Elastic Block Store (Amazon EBS) yang dibuat menggunakan fitur penyimpanan persisten Kubernetes. Mode Otomatis EKS juga tidak mengelola Elastic Load Balancers (ELB). Amazon EKS Auto Mode mengotomatiskan tugas rutin untuk sumber daya ini.

Keamanan penyimpanan

- AWS merekomendasikan agar Anda mengaktifkan enkripsi untuk Volume EBS yang disediakan oleh fitur penyimpanan persisten Kubernetes. Untuk informasi selengkapnya, lihat [the section called “Buat StorageClass”](#).
- Enkripsi saat istirahat menggunakan AWS KMS
- Anda dapat mengonfigurasi AWS akun Anda untuk menerapkan enkripsi volume EBS baru dan salinan snapshot yang Anda buat. Untuk informasi selengkapnya, lihat [Mengaktifkan enkripsi Amazon EBS secara default](#) di Panduan Pengguna Amazon EBS.
- Untuk informasi selengkapnya, lihat [Keamanan di Amazon EBS](#).

Keamanan penyeimbang beban

- Konfigurasi otomatis Elastic Load Balancers
- Manajemen sertifikat SSL/TLS melalui integrasi Certificate Manager AWS

- Otomatisasi grup keamanan untuk kontrol akses penyeimbang beban
- Untuk informasi selengkapnya, lihat [Keamanan di Elastic Load Balancing](#).

Praktik terbaik keamanan

Bagian berikut menjelaskan praktik terbaik keamanan untuk Amazon EKS Auto Mode.

- Tinjau kebijakan AWS IAM dan entri akses EKS secara teratur.
- Menerapkan pola akses hak istimewa terkecil untuk beban kerja.
- Pantau aktivitas cluster melalui AWS CloudTrail dan Amazon CloudWatch. Untuk informasi selengkapnya, silakan lihat [the section called “AWS CloudTrail”](#) dan [the section called “Amazon CloudWatch”](#).
- Gunakan AWS Security Hub untuk penilaian postur keamanan.
- Terapkan standar keamanan pod yang sesuai untuk beban kerja Anda.

Identity and access management untuk Amazon EKS

AWS Identity and Access Management (IAM) adalah AWS layanan yang membantu administrator mengontrol akses ke sumber daya dengan aman. AWS Administrator IAM mengendalikan siapa yang dapat diautentikasi (masuk) dan diotorisasi (memiliki izin) untuk menggunakan sumber daya Amazon EKS. IAM adalah AWS layanan yang dapat Anda gunakan tanpa biaya tambahan.

Audiens

Cara Anda menggunakan AWS Identity and Access Management (IAM) berbeda, tergantung pada pekerjaan yang Anda lakukan di Amazon EKS.

Pengguna layanan – Jika Anda menggunakan layanan Amazon EKS untuk melakukan tugas, maka administrator Anda akan memberikan kredensial dan izin yang dibutuhkan. Saat Anda menggunakan lebih banyak fitur di Amazon EKS untuk melakukan pekerjaan, Anda mungkin memerlukan izin tambahan. Memahami bagaimana akses dikelola dapat membantu Anda untuk meminta izin yang tepat dari administrator Anda. Jika Anda tidak dapat mengakses fitur di Amazon EKS, lihat [the section called “Pemecahan Masalah”](#).

Administrator layanan - Jika Anda bertanggung jawab atas sumber daya Amazon EKS di perusahaan Anda, Anda mungkin memiliki akses penuh ke Amazon EKS. Tugas Anda adalah menentukan fitur dan sumber daya Amazon EKS mana yang harus diakses pengguna layanan Anda. Kemudian, Anda

harus mengirimkan permintaan kepada administrator IAM untuk mengubah izin pengguna layanan Anda. Tinjau informasi di halaman ini untuk memahami konsep dasar IAM. Untuk mempelajari selengkapnya tentang bagaimana perusahaan Anda dapat menggunakan IAM dengan Amazon EKS, lihat [the section called “Amazon EKS dan IAM”](#).

Administrator IAM - Jika Anda seorang administrator IAM, Anda mungkin ingin mempelajari detail tentang cara menulis kebijakan untuk mengelola akses ke Amazon EKS. Untuk melihat contoh kebijakan berbasis identitas Amazon EKS yang dapat Anda gunakan di IAM, lihat [the section called “Kebijakan berbasis identitas”](#).

Mengautentikasi dengan identitas

Otentikasi adalah cara Anda masuk AWS menggunakan kredensi identitas Anda. Anda harus diautentikasi (masuk ke AWS) sebagai pengguna root AWS akun, sebagai pengguna IAM, atau dengan mengambil peran IAM.

Anda dapat masuk AWS sebagai identitas federasi dengan menggunakan kredensial yang disediakan melalui sumber identitas. AWS Pengguna IAM Identity Center (IAM Identity Center), autentikasi masuk tunggal perusahaan Anda, dan kredensi Google atau Facebook Anda adalah contoh identitas federasi. Saat Anda masuk sebagai identitas terfederasi, administrator Anda sebelumnya menyiapkan federasi identitas menggunakan peran IAM. Ketika Anda mengakses AWS dengan menggunakan federasi, Anda secara tidak langsung mengambil peran.

Bergantung pada jenis pengguna Anda, Anda dapat masuk ke AWS Management Console atau portal AWS akses. Untuk informasi selengkapnya tentang masuk AWS, lihat [Cara masuk ke AWS akun Anda](#) di Panduan Pengguna AWS Masuk.

Jika Anda mengakses AWS secara terprogram, AWS sediakan kit pengembangan perangkat lunak (SDK) dan antarmuka baris perintah (CLI) untuk menandatangani permintaan Anda secara kriptografis dengan menggunakan kredensial Anda. Jika Anda tidak menggunakan AWS alat, Anda harus menandatangani permintaan sendiri. Untuk informasi selengkapnya tentang penggunaan metode yang disarankan untuk menandatangani permintaan sendiri, lihat [Menandatangani permintaan AWS API](#) di Panduan Pengguna IAM.

Apa pun metode autentikasi yang digunakan, Anda mungkin diminta untuk menyediakan informasi keamanan tambahan. Misalnya, AWS merekomendasikan agar Anda menggunakan otentikasi multi-faktor (MFA) untuk meningkatkan keamanan akun Anda. Untuk mempelajari lebih lanjut, lihat [Autentikasi multi-faktor](#) di Panduan Pengguna Pusat AWS Identitas IAM dan [Menggunakan otentikasi multi-faktor \(MFA\)](#) di Panduan Pengguna IAM. AWS

AWS pengguna root akun

Saat Anda membuat AWS akun, Anda mulai dengan satu identitas masuk yang memiliki akses lengkap ke semua AWS layanan dan sumber daya di akun. Identitas ini disebut pengguna root AWS akun dan diakses dengan masuk dengan alamat email dan kata sandi yang Anda gunakan untuk membuat akun. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari Anda. Lindungi kredensial pengguna root Anda dan gunakan kredensial tersebut untuk melakukan tugas yang hanya dapat dilakukan pengguna root. Untuk daftar lengkap tugas yang mengharuskan Anda masuk sebagai pengguna root, lihat [Tugas yang memerlukan kredensial pengguna root](#) dalam Panduan Pengguna IAM.

Pengguna dan grup IAM

[Pengguna IAM](#) adalah identitas dalam AWS akun Anda yang memiliki izin khusus untuk satu orang atau aplikasi. Jika memungkinkan, kami merekomendasikan untuk mengandalkan kredensial sementara, bukan membuat pengguna IAM yang memiliki kredensial jangka panjang seperti kata sandi dan kunci akses. Namun, jika Anda memiliki kasus penggunaan tertentu yang memerlukan kredensial jangka panjang dengan pengguna IAM, kami merekomendasikan Anda merotasi kunci akses. Untuk informasi selengkapnya, lihat [Merotasi kunci akses secara teratur untuk kasus penggunaan yang memerlukan kredensial jangka panjang](#) dalam Panduan Pengguna IAM.

[Grup IAM](#) adalah identitas yang menentukan sekumpulan pengguna IAM. Anda tidak dapat masuk sebagai grup. Anda dapat menggunakan grup untuk menentukan izin bagi beberapa pengguna sekaligus. Grup mempermudah manajemen izin untuk sejumlah besar pengguna sekaligus. Misalnya, Anda dapat meminta kelompok untuk menyebutkan IAMAdmins dan memberikan izin kepada grup tersebut untuk mengelola sumber daya IAM.

Pengguna berbeda dari peran. Pengguna secara unik terkait dengan satu orang atau aplikasi, tetapi peran dimaksudkan untuk dapat digunakan oleh siapa pun yang membutuhkannya. Pengguna memiliki kredensial jangka panjang permanen, tetapi peran memberikan kredensial sementara. Untuk mempelajari selengkapnya, lihat [Kapan harus membuat pengguna IAM \(bukan peran\)](#) dalam Panduan Pengguna IAM.

Peran IAM

[Peran IAM](#) adalah identitas dalam AWS akun Anda yang memiliki izin khusus. Peran ini mirip dengan pengguna IAM, tetapi tidak terkait dengan orang tertentu. Anda dapat mengambil peran IAM untuk sementara AWS Management Console dengan [beralih peran](#). Anda dapat mengambil peran

dengan memanggil operasi AWS CLI atau AWS API atau dengan menggunakan URL khusus. Untuk informasi selengkapnya tentang cara menggunakan peran, lihat [Menggunakan peran IAM](#) dalam Panduan Pengguna IAM.

Peran IAM dengan kredensial sementara berguna dalam situasi berikut:

- Akses pengguna terfederasi – Untuk menetapkan izin ke identitas terfederasi, Anda membuat peran dan menentukan izin untuk peran tersebut. Ketika identitas terfederasi mengautentikasi, identitas tersebut terhubung dengan peran dan diberi izin yang ditentukan oleh peran. Untuk informasi tentang peran untuk federasi, lihat [Membuat peran untuk Penyedia Identitas pihak ketiga](#) dalam Panduan Pengguna IAM. Jika menggunakan Pusat Identitas IAM, Anda harus mengonfigurasi set izin. Untuk mengontrol apa yang dapat diakses identitas Anda setelah identitas tersebut diautentikasi, Pusat Identitas IAM akan mengorelasikan set izin ke peran dalam IAM. Untuk informasi tentang set izin, lihat [Set izin](#) di Panduan Pengguna Pusat AWS Identitas IAM.
- Izin pengguna IAM sementara – Pengguna atau peran IAM dapat mengambil peran IAM guna mendapatkan berbagai izin secara sementara untuk tugas tertentu.
- Akses lintas akun – Anda dapat menggunakan peran IAM untuk mengizinkan seseorang (prinsipal tepercaya) di akun lain untuk mengakses sumber daya di akun Anda. Peran adalah cara utama untuk memberikan akses lintas akun. Namun, dengan beberapa AWS layanan, Anda dapat melampirkan kebijakan langsung ke sumber daya (alih-alih menggunakan peran sebagai proxy). Untuk mempelajari perbedaan antara peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat [Akses sumber daya lintas akun di IAM](#) dalam Panduan Pengguna IAM.
- Akses lintas layanan — Beberapa AWS layanan menggunakan fitur di AWS layanan lain. Misalnya, saat Anda melakukan panggilan dalam suatu layanan, biasanya layanan tersebut menjalankan aplikasi di Amazon EC2 atau menyimpan objek di Amazon S3. Layanan mungkin melakukan ini menggunakan izin kepala panggilan, menggunakan peran layanan, atau menggunakan peran terkait layanan.
- Sesi akses teruskan (FAS) — Saat Anda menggunakan pengguna IAM atau peran untuk melakukan tindakan AWS, Anda dianggap sebagai prinsipal. Ketika Anda menggunakan beberapa layanan, Anda mungkin melakukan sebuah tindakan yang kemudian menginisiasi tindakan lain di layanan yang berbeda. FAS menggunakan izin dari prinsipal yang memanggil AWS layanan, dikombinasikan dengan layanan yang meminta untuk membuat permintaan ke AWS layanan hilir. Permintaan FAS hanya dibuat ketika layanan menerima permintaan yang memerlukan interaksi dengan AWS layanan atau sumber daya lain untuk diselesaikan. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan ketika mengajukan permintaan FAS, lihat [Sesi akses maju](#).

- Peran layanan – Peran layanan adalah [peran IAM](#) yang dijalankan oleh layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, mengubah, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat [Membuat peran untuk mendelegasikan izin ke AWS layanan di Panduan Pengguna IAM](#).
- Peran terkait layanan — Peran terkait layanan adalah jenis peran layanan yang ditautkan ke layanan. AWS Layanan tersebut dapat menjalankan peran untuk melakukan tindakan atas nama Anda. Peran terkait layanan muncul di AWS akun Anda dan dimiliki oleh layanan. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.
- Aplikasi yang berjalan di Amazon EC2 — Anda dapat menggunakan peran IAM untuk mengelola kredensi sementara untuk aplikasi yang berjalan pada EC2 instance dan membuat permintaan AWS CLI atau API. AWS Ini lebih baik untuk menyimpan kunci akses dalam EC2 instance. Untuk menetapkan AWS peran ke EC2 instance dan membuatnya tersedia untuk semua aplikasinya, Anda membuat profil instance yang dilampirkan ke instance. Profil instance berisi peran dan memungkinkan program yang berjalan pada EC2 instance untuk mendapatkan kredensi sementara. Untuk informasi selengkapnya, lihat [Menggunakan peran IAM untuk memberikan izin ke aplikasi yang berjalan di EC2 instans Amazon di Panduan Pengguna IAM](#).

Untuk mempelajari apakah kita harus menggunakan peran IAM atau pengguna IAM, lihat [Kapan harus membuat peran IAM \(bukan pengguna\)](#) dalam Panduan Pengguna IAM.

Mengelola akses menggunakan kebijakan

Anda mengontrol akses AWS dengan membuat kebijakan dan melampirkannya ke AWS identitas atau sumber daya. Kebijakan adalah objek AWS yang, ketika dikaitkan dengan identitas atau sumber daya, menentukan izinnya. AWS mengevaluasi kebijakan ini ketika prinsipal (pengguna, pengguna root, atau sesi peran) membuat permintaan. Izin dalam kebijakan menentukan apakah permintaan diizinkan atau ditolak. Sebagian besar kebijakan disimpan AWS sebagai dokumen JSON. Untuk informasi selengkapnya tentang struktur dan isi dokumen kebijakan JSON, lihat [Gambaran umum kebijakan JSON](#) dalam Panduan Pengguna IAM.

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Yaitu, principal dapat melakukan tindakan pada suatu sumber daya, dan dalam suatu syarat.

Secara default, pengguna dan peran tidak memiliki izin. Untuk memberikan izin kepada pengguna untuk melakukan tindakan di sumber daya yang mereka perlukan, administrator IAM dapat membuat

kebijakan IAM. Administrator kemudian dapat menambahkan kebijakan IAM ke peran, dan pengguna dapat mengambil peran.

Kebijakan IAM mendefinisikan izin untuk suatu tindakan terlepas dari metode yang Anda gunakan untuk melakukan operasinya. Misalnya, anggaplah Anda memiliki kebijakan yang mengizinkan tindakan `iam:GetRole`. Pengguna dengan kebijakan itu bisa mendapatkan informasi peran dari AWS Management Console, AWS CLI, atau API. AWS

Kebijakan berbasis identitas

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke sebuah identitas, seperti pengguna IAM, grup pengguna IAM, atau peran IAM. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan oleh pengguna dan peran, di sumber daya mana, dan berdasarkan kondisi seperti apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat [Membuat kebijakan IAM](#) dalam Panduan Pengguna IAM.

Kebijakan berbasis identitas dapat dikategorikan lebih lanjut sebagai kebijakan inline atau kebijakan yang dikelola. Kebijakan inline disematkan langsung ke satu pengguna, grup, atau peran. Kebijakan terkelola adalah kebijakan mandiri yang dapat Anda lampirkan ke beberapa pengguna, grup, dan peran di AWS akun Anda. Kebijakan AWS terkelola mencakup kebijakan terkelola dan kebijakan yang dikelola pelanggan. Untuk mempelajari cara memilih antara kebijakan yang dikelola atau kebijakan inline, lihat [Memilih antara kebijakan yang dikelola dan kebijakan inline](#) dalam Panduan Pengguna IAM.

Kebijakan berbasis sumber daya

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan kepercayaan peran IAM dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Untuk sumber daya tempat kebijakan dilampirkan, kebijakan menentukan tindakan apa yang dapat dilakukan oleh prinsipal tertentu pada sumber daya tersebut dan dalam kondisi apa. Anda harus [menentukan prinsipal](#) dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup akun, pengguna, peran, pengguna federasi, atau layanan. AWS

Kebijakan berbasis sumber daya merupakan kebijakan inline yang terletak di layanan tersebut. Anda tidak dapat menggunakan kebijakan AWS terkelola dari IAM dalam kebijakan berbasis sumber daya.

Daftar kontrol akses (ACLs)

Access control lists (ACLs) mengontrol prinsipal mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACLs mirip dengan kebijakan berbasis sumber daya, meskipun mereka tidak menggunakan format dokumen kebijakan JSON.

Amazon S3, AWS WAF, dan Amazon VPC adalah contoh layanan yang mendukung ACLs.

Untuk mempelajari selengkapnya ACLs, lihat [Ringkasan daftar kontrol akses \(ACL\)](#) di Panduan Pengembang Layanan Penyimpanan Sederhana Amazon.

Jenis-jenis kebijakan lain

AWS mendukung jenis kebijakan tambahan yang kurang umum. Jenis-jenis kebijakan ini dapat mengatur izin maksimum yang diberikan kepada Anda oleh jenis kebijakan yang lebih umum.

- **Batasan izin** – Batasan izin adalah fitur lanjutan tempat Anda mengatur izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas ke entitas IAM (pengguna IAM atau peran IAM). Anda dapat menetapkan batasan izin untuk suatu entitas. Izin yang dihasilkan adalah persimpangan kebijakan berbasis identitas entitas dan batas izinnya. Kebijakan berbasis sumber daya yang menentukan pengguna atau peran dalam bidang `Principal` tidak dibatasi oleh batasan izin. Penolakan eksplisit dalam salah satu kebijakan ini akan menggantikan pemberian izin. Untuk informasi selengkapnya tentang batasan izin, lihat [Batasan izin untuk entitas IAM](#) dalam Panduan Pengguna IAM.
- **Kebijakan kontrol layanan (SCPs)** — SCPs adalah kebijakan JSON yang menentukan izin maksimum untuk organisasi atau unit organisasi (OU) di Organizations AWS . AWS Organizations adalah layanan untuk mengelompokkan dan mengelola beberapa AWS akun yang dimiliki bisnis Anda secara terpusat. Jika Anda mengaktifkan semua fitur dalam suatu organisasi, maka Anda dapat menerapkan kebijakan kontrol layanan (SCPs) ke salah satu atau semua akun Anda. SCP membatasi izin untuk entitas di akun anggota, termasuk setiap pengguna root AWS akun. Untuk informasi selengkapnya tentang Organizations dan SCPs, lihat [Kebijakan kontrol layanan](#) di Panduan Pengguna AWS Organizations.
- **Kebijakan sesi** – Kebijakan sesi adalah kebijakan lanjutan yang Anda berikan sebagai parameter ketika Anda membuat sesi sementara secara programatis untuk peran atau pengguna terfederasi. Izin sesi yang dihasilkan adalah persimpangan kebijakan berbasis identitas pengguna atau peran dan kebijakan sesi. Izin juga bisa datang dari kebijakan berbasis sumber daya. Penolakan eksplisit dalam salah satu kebijakan ini akan menggantikan pemberian izin. Untuk informasi selengkapnya, lihat [Kebijakan sesi](#) dalam Panduan Pengguna IAM.

Berbagai jenis kebijakan

Ketika beberapa jenis kebijakan berlaku pada suatu permintaan, izin yang dihasilkan lebih rumit untuk dipahami. Untuk mempelajari cara AWS menentukan apakah akan mengizinkan permintaan saat beberapa jenis kebijakan terlibat, lihat [Logika evaluasi kebijakan](#) di Panduan Pengguna IAM.

Bagaimana cara Amazon EKS bekerja sama dengan IAM

Sebelum Anda menggunakan IAM untuk mengelola akses ke Amazon EKS, Anda harus memahami fitur-fitur IAM apa yang tersedia untuk digunakan dengan Amazon EKS. Untuk mendapatkan tampilan tingkat tinggi tentang cara Amazon EKS dan AWS layanan lainnya bekerja dengan IAM, lihat [AWS layanan yang bekerja dengan IAM di Panduan Pengguna IAM](#).

Topik

- [Kebijakan berbasis identitas Amazon EKS](#)
- [kebijakan berbasis sumber daya Amazon EKS](#)
- [Otorisasi berdasarkan tanda Amazon EKS](#)
- [IAM role Amazon EKS](#)

Kebijakan berbasis identitas Amazon EKS

Dengan kebijakan berbasis identitas IAM, Anda dapat menentukan apakah tindakan dan sumber daya diizinkan atau ditolak, serta persyaratan terkait diizinkan atau ditolak-nya tindakan tersebut. Amazon EKS mendukung tindakan, sumber daya, dan kunci syarat tertentu. Untuk mempelajari semua elemen yang Anda gunakan dalam kebijakan JSON, lihat [Referensi elemen kebijakan IAM JSON](#) dalam Panduan Pengguna IAM.

Tindakan

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Artinya, prinsipal manakah yang dapat melakukan tindakan pada sumber daya apa, dan dengan kondisi apa.

Elemen `Action` dari kebijakan JSON menjelaskan tindakan yang dapat Anda gunakan untuk mengizinkan atau menolak akses dalam sebuah kebijakan. Tindakan kebijakan biasanya memiliki nama yang sama dengan operasi AWS API terkait. Ada beberapa pengecualian, seperti tindakan khusus izin yang tidak memiliki operasi API yang cocok. Ada juga beberapa operasi yang

memerlukan beberapa tindakan dalam suatu kebijakan. Tindakan tambahan ini disebut tindakan dependen.

Sertakan tindakan dalam kebijakan untuk memberikan izin pelaksanaan operasi yang terkait.

Tindakan kebijakan di Amazon EKS menggunakan prefiks berikut sebelum tindakan: `eks:`. Misalnya, untuk memberikan izin kepada seseorang agar mendapatkan informasi deskriptif tentang kluster Amazon EKS, Anda menyertakan tindakan `DescribeCluster` dalam kebijakan mereka. Pernyataan kebijakan harus memuat elemen `Action` atau `NotAction`.

Untuk menetapkan beberapa tindakan dalam satu pernyataan, pisahkan dengan koma seperti berikut:

```
"Action": ["eks:action1", "eks:action2"]
```

Anda dapat menentukan beberapa tindakan menggunakan wildcard (*). Sebagai contoh, untuk menentukan semua tindakan yang dimulai dengan kata `Describe`, sertakan tindakan berikut:

```
"Action": "eks:Describe*"
```

Untuk melihat daftar tindakan Amazon EKS, lihat [Tindakan yang ditentukan oleh Amazon Elastic Kubernetes Service di Referensi Otorisasi Layanan](#).

Sumber daya

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Artinya, prinsipal manakah yang dapat melakukan tindakan pada sumber daya apa, dan dengan kondisi apa.

Elemen kebijakan JSON `Resource` menentukan objek yang menjadi target penerapan tindakan. Pernyataan harus menyertakan elemen `Resource` atau `NotResource`. Praktik terbaiknya, tentukan sumber daya menggunakan [Amazon Resource Name \(ARN\)](#). Anda dapat melakukan ini untuk tindakan yang mendukung jenis sumber daya tertentu, yang dikenal sebagai izin tingkat sumber daya.

Untuk tindakan yang tidak mendukung izin tingkat sumber daya, seperti operasi daftar, gunakan wildcard (*) untuk menunjukkan bahwa pernyataan tersebut berlaku untuk semua sumber daya.

```
"Resource": "*"
```

Sumber daya cluster Amazon EKS memiliki ARN berikut.

```
arn:aws: eks:region-code:account-id:cluster/cluster-name
```

Untuk informasi selengkapnya tentang format ARNs, lihat [Amazon resource names \(ARNs\) dan ruang nama AWS layanan](#).

Misalnya, untuk menentukan cluster dengan nama *my-cluster* dalam pernyataan Anda, gunakan ARN berikut:

```
"Resource": "arn:aws: eks:region-code:111122223333:cluster/my-cluster"
```

Untuk menentukan semua cluster milik akun dan AWS Wilayah tertentu, gunakan wildcard (*):

```
"Resource": "arn:aws: eks:region-code:111122223333:cluster/*"
```

Beberapa tindakan Amazon EKS, seperti untuk membuat sumber daya, tidak dapat dilakukan pada sumber daya tertentu. Dalam kasus tersebut, Anda harus menggunakan wildcard (*).

```
"Resource": ""
```

Untuk melihat daftar jenis sumber daya Amazon EKS dan jenisnya ARNs, lihat Sumber [daya yang ditentukan oleh Amazon Elastic Kubernetes Service dalam Referensi Otorisasi Layanan](#). Untuk mempelajari tindakan mana yang dapat Anda tentukan ARN dari setiap resource, lihat [Tindakan yang ditentukan oleh Amazon Elastic Kubernetes Service](#).

Kunci syarat

Amazon EKS menentukan set kunci syaratnya sendiri dan juga mendukung penggunaan beberapa kunci syarat global. Untuk melihat semua kunci kondisi AWS global, lihat [Kunci Konteks Kondisi AWS Global](#) di Panduan Pengguna IAM.

Anda dapat mengatur kunci syarat ketika mengaitkan penyedia OpenID Connect ke klaster Anda. Untuk informasi selengkapnya, lihat [the section called “Contoh kebijakan IAM”](#).

Semua EC2 tindakan Amazon mendukung kunci `aws:RequestedRegion` dan `ec2:Region` kondisi. Untuk informasi selengkapnya, lihat [Contoh: Membatasi Akses ke AWS Wilayah Tertentu](#).

Untuk daftar kunci kondisi Amazon EKS, lihat [Ketentuan yang ditentukan oleh Amazon Elastic Kubernetes Service di Referensi Otorisasi Layanan](#). Untuk mempelajari tindakan dan sumber daya yang dapat digunakan untuk menggunakan kunci kondisi, lihat [Tindakan yang ditentukan oleh Amazon Elastic Kubernetes Service](#).

Contoh

Untuk melihat contoh kebijakan berbasis identitas Amazon EKS, lihat [the section called “Kebijakan berbasis identitas”](#).

Saat Anda membuat klaster Amazon EKS, [prinsipal IAM](#) yang membuat klaster secara otomatis diberikan `system:masters` izin dalam konfigurasi kontrol akses berbasis peran (RBAC) klaster di bidang kontrol Amazon EKS. Prinsipal ini tidak muncul dalam konfigurasi yang terlihat, jadi pastikan untuk melacak prinsipal mana yang awalnya membuat cluster. Untuk memberikan prinsipal IAM tambahan kemampuan untuk berinteraksi dengan klaster Anda, edit bagian `aws-auth` ConfigMap dalam Kubernetes dan buat Kubernetes `rolebinding` atau `clusterrolebinding` dengan nama yang Anda tentukan di dalamnya. `group aws-auth` ConfigMap

Untuk informasi lebih lanjut tentang bekerja dengan ConfigMap, lihat [the section called “Akses API Kubernetes”](#).

kebijakan berbasis sumber daya Amazon EKS

Amazon EKS tidak mendukung kebijakan berbasis sumber daya.

Otorisasi berdasarkan tanda Amazon EKS

Anda dapat melampirkan tanda ke sumber daya Amazon EKS atau meneruskan tanda dalam sebuah permintaan ke Amazon EKS. Untuk mengendalikan akses berdasarkan tanda, Anda dapat memberikan informasi tentang tanda di [elemen syarat](#) kebijakan menggunakan kunci kondisi `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, atau `aws:TagKeys`. Untuk informasi selengkapnya tentang penandaan sumber daya Amazon EKS, lihat [the section called “Pemberian tag pada sumber daya Anda”](#). Untuk informasi selengkapnya tentang tindakan yang dapat Anda gunakan dengan tag dalam kunci kondisi, lihat [Tindakan yang ditentukan oleh Amazon EKS](#) di [Referensi Otorisasi Layanan](#).

IAM role Amazon EKS

[Peran IAM](#) adalah entitas dalam AWS akun Anda yang memiliki izin tertentu.

Menggunakan kredensial sementara dengan Amazon EKS

Anda dapat menggunakan kredensial sementara untuk masuk dengan gabungan, menjalankan IAM role, atau menjalankan peran lintas akun. Anda memperoleh kredensi keamanan sementara dengan memanggil operasi AWS STS API seperti [AssumeRole](#) atau [GetFederationToken](#).

Amazon EKS mendukung penggunaan kredensial sementara.

Peran terkait layanan

[Peran terkait AWS layanan](#) memungkinkan layanan mengakses sumber daya di layanan lain untuk menyelesaikan tindakan atas nama Anda. Peran terkait layanan muncul di akun IAM Anda dan dimiliki oleh layanan tersebut. Administrator dapat melihat tetapi tidak dapat mengedit izin untuk peran terkait layanan.

Amazon EKS mendukung peran terkait-layanan. Untuk informasi selengkapnya tentang cara membuat atau mengelola peran terkait layanan Amazon EKS, lihat [the section called “Peran terkait layanan”](#).

Peran layanan

Fitur ini memungkinkan layanan untuk menerima [peran layanan](#) atas nama Anda. Peran ini mengizinkan layanan untuk mengakses sumber daya di layanan lain untuk menyelesaikan tindakan atas nama Anda. Peran layanan muncul di akun IAM Anda dan dimiliki oleh akun tersebut. Ini berarti administrator IAM dapat mengubah izin untuk peran ini. Namun, hal itu dapat merusak fungsionalitas layanan.

Amazon EKS mendukung peran layanan. Untuk informasi selengkapnya, lihat [the section called “IAM role kluster”](#) dan [the section called “IAM role simpul”](#).

Memilih IAM role di Amazon EKS

Saat Anda membuat sumber daya kluster di Amazon EKS, Anda harus memilih peran untuk memungkinkan Amazon EKS mengakses beberapa AWS sumber daya lain atas nama Anda. Jika sebelumnya Anda telah membuat peran layanan, maka Amazon EKS akan memberi Anda daftar peran untuk dipilih. Penting untuk memilih peran yang memiliki kebijakan terkelola Amazon EKS yang melekat padanya. Lihat informasi yang lebih lengkap di [the section called “Periksa apakah peran kluster sudah ada”](#) dan [the section called “Periksa apakah peran simpul sudah ada”](#).

Contoh kebijakan berbasis identitas Amazon EKS

Secara default, pengguna dan peran IAM tidak memiliki izin untuk membuat atau memodifikasi sumber daya Amazon EKS. Mereka juga tidak dapat melakukan tugas menggunakan AWS Management Console, AWS CLI, atau AWS API. Administrator IAM harus membuat kebijakan IAM yang memberikan izin kepada pengguna dan peran untuk melakukan operasi API tertentu pada sumber daya yang diperlukan. Administrator kemudian harus melampirkan kebijakan tersebut ke pengguna IAM atau grup yang memerlukan izin tersebut.

Untuk mempelajari cara membuat kebijakan berbasis identitas IAM menggunakan contoh dokumen kebijakan JSON ini, lihat [Membuat kebijakan di tab JSON](#) dalam Panduan Pengguna IAM.

Saat Anda membuat klaster Amazon EKS, [prinsipal IAM](#) yang membuat klaster secara otomatis diberikan `system:masters` izin dalam konfigurasi kontrol akses berbasis peran (RBAC) klaster di bidang kontrol Amazon EKS. Prinsipal ini tidak muncul dalam konfigurasi yang terlihat, jadi pastikan untuk melacak prinsipal mana yang awalnya membuat cluster. Untuk memberikan prinsipal IAM tambahan kemampuan untuk berinteraksi dengan klaster Anda, edit bagian `aws-auth` ConfigMap dalam Kubernetes dan buat Kubernetes `rolebinding` atau `clusterrolebinding` dengan nama yang Anda tentukan di dalamnya. `group aws-auth` ConfigMap

Untuk informasi lebih lanjut tentang bekerja dengan ConfigMap, lihat [the section called “Akses API Kubernetes”](#).

Topik

- [Praktik terbaik kebijakan](#)
- [Menggunakan konsol Amazon EKS](#)
- [Izinkan pengguna IAM untuk melihat izin mereka sendiri](#)
- [Buat klaster Kubernetes di Cloud AWS](#)
- [Buat klaster Kubernetes lokal di Outpost](#)
- [Perbarui klaster Kubernetes](#)
- [Buat daftar atau deskripsikan semua klaster](#)

Praktik terbaik kebijakan

Kebijakan berbasis identitas menentukan apakah seseorang dapat membuat, mengakses, atau menghapus sumber daya Amazon EKS di akun Anda. Tindakan ini dapat menimbulkan biaya untuk AWS akun Anda. Ketika Anda membuat atau mengedit kebijakan berbasis identitas, ikuti panduan dan rekomendasi ini:

- Mulailah dengan kebijakan AWS terkelola dan beralih ke izin hak istimewa paling sedikit — Untuk mulai memberikan izin kepada pengguna dan beban kerja Anda, gunakan kebijakan AWS terkelola yang memberikan izin untuk banyak kasus penggunaan umum. Mereka tersedia di AWS akun Anda. Kami menyarankan Anda mengurangi izin lebih lanjut dengan menentukan kebijakan yang dikelola AWS pelanggan yang khusus untuk kasus penggunaan Anda. Untuk informasi selengkapnya, lihat [Kebijakan yang dikelola AWS](#) atau [Kebijakan yang dikelola AWS untuk fungsi tugas](#) dalam Panduan Pengguna IAM.

- Menerapkan izin dengan hak akses paling rendah – Ketika Anda menetapkan izin dengan kebijakan IAM, hanya berikan izin yang diperlukan untuk melakukan tugas. Anda melakukannya dengan mendefinisikan tindakan yang dapat diambil pada sumber daya tertentu dalam kondisi tertentu, yang juga dikenal sebagai izin dengan hak akses paling rendah. Untuk informasi selengkapnya tentang cara menggunakan IAM untuk mengajukan izin, lihat [Kebijakan dan izin dalam IAM](#) dalam Panduan Pengguna IAM.
- Gunakan kondisi dalam kebijakan IAM untuk membatasi akses lebih lanjut – Anda dapat menambahkan suatu kondisi ke kebijakan Anda untuk membatasi akses ke tindakan dan sumber daya. Sebagai contoh, Anda dapat menulis kondisi kebijakan untuk menentukan bahwa semua permintaan harus dikirim menggunakan SSL. Anda juga dapat menggunakan ketentuan untuk memberikan akses ke tindakan layanan jika digunakan melalui AWS layanan tertentu, seperti AWS CloudFormation. Untuk informasi selengkapnya, lihat [Elemen kebijakan JSON IAM: Kondisi](#) dalam Panduan Pengguna IAM.
- Gunakan IAM Access Analyzer untuk memvalidasi kebijakan IAM Anda untuk memastikan izin yang aman dan fungsional – IAM Access Analyzer memvalidasi kebijakan baru dan yang sudah ada sehingga kebijakan tersebut mematuhi bahasa kebijakan IAM (JSON) dan praktik terbaik IAM. IAM Access Analyzer menyediakan lebih dari 100 pemeriksaan kebijakan dan rekomendasi yang dapat ditindaklanjuti untuk membantu Anda membuat kebijakan yang aman dan fungsional. Untuk informasi selengkapnya, lihat [Validasi kebijakan IAM Access Analyzer](#) dalam Panduan Pengguna IAM.
- Memerlukan otentikasi multi-faktor (MFA) - Jika Anda memiliki skenario yang mengharuskan pengguna IAM atau pengguna root di akun Anda AWS, aktifkan MFA untuk keamanan tambahan. Untuk meminta MFA ketika operasi API dipanggil, tambahkan kondisi MFA pada kebijakan Anda. Untuk informasi selengkapnya, lihat [Mengonfigurasi akses API yang dilindungi MFA](#) dalam Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang praktik terbaik dalam IAM, lihat [Praktik terbaik keamanan dalam IAM](#) dalam Panduan Pengguna IAM.

Menggunakan konsol Amazon EKS

Untuk mengakses konsol Amazon EKS, [prinsipal IAM](#), harus memiliki set izin minimum. Izin ini memungkinkan kepala sekolah untuk membuat daftar dan melihat detail tentang sumber daya Amazon EKS di AWS akun Anda. Jika Anda membuat kebijakan berbasis identitas yang lebih ketat daripada izin minimum yang diperlukan, konsol tidak akan berfungsi sebagaimana dimaksud untuk prinsipal dengan kebijakan yang dilampirkan padanya.

Untuk memastikan bahwa prinsipal IAM Anda masih dapat menggunakan konsol Amazon EKS, buat kebijakan dengan nama unik Anda sendiri, seperti. `AmazonEKSAAdminPolicy` Lampirkan kebijakan ke kepala sekolah. Untuk informasi lebih lanjut, lihat [Menambahkan dan menghapus izin identitas IAM](#) dalam Panduan Pengguna IAM.

 Important

Contoh kebijakan berikut memungkinkan prinsipal untuk melihat informasi pada tab Konfigurasi di konsol. Untuk melihat informasi pada tab Ikhtisar dan Sumber Daya di AWS Management Console, prinsipal juga membutuhkan izin Kubernetes. Untuk informasi selengkapnya, lihat [the section called "Izin yang diperlukan"](#).

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "eks:*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "eks.amazonaws.com"
        }
      }
    }
  ]
}
```

Anda tidak perlu mengizinkan izin konsol minimum untuk prinsipal yang melakukan panggilan hanya ke CLI AWS atau API. AWS Sebagai gantinya, izinkan akses hanya ke tindakan yang cocok dengan operasi API yang Anda coba lakukan.

Izinkan pengguna IAM untuk melihat izin mereka sendiri

Contoh ini menunjukkan cara membuat kebijakan yang mengizinkan pengguna IAM melihat kebijakan inline dan terkelola yang dilampirkan ke identitas pengguna mereka. Kebijakan ini mencakup izin untuk menyelesaikan tindakan ini di konsol atau menggunakan CLI AWS atau API secara terprogram. AWS

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetUserPolicy",
        "iam:ListGroupsForUser",
        "iam:ListAttachedUserPolicies",
        "iam:ListUserPolicies",
        "iam:GetUser"
      ],
      "Resource": ["arn:aws:iam::*:user/${aws:username}"]
    },
    {
      "Sid": "NavigateInConsole",
      "Effect": "Allow",
      "Action": [
        "iam:GetGroupPolicy",
        "iam:GetPolicyVersion",
        "iam:GetPolicy",
        "iam:ListAttachedGroupPolicies",
        "iam:ListGroupPolicies",
        "iam:ListPolicyVersions",
        "iam:ListPolicies",
        "iam:ListUsers"
      ],
      "Resource": "*"
    }
  ]
}
```

Buat kluster Kubernetes di Cloud AWS

Kebijakan contoh ini mencakup izin minimum yang diperlukan untuk membuat kluster Amazon EKS yang diberi nama *my-cluster* di *us-west-2* AWS Wilayah. Anda dapat mengganti AWS Region dengan AWS Region tempat Anda ingin membuat kluster. Jika Anda melihat peringatan yang mengatakan Tindakan dalam kebijakan Anda tidak mendukung izin tingkat sumber daya dan mengharuskan Anda untuk memilih AWS Management Console, tindakan tersebut dapat All resources diabaikan dengan aman. Jika akun Anda sudah memiliki *AWSServiceRoleForAmazonEKS* peran, Anda dapat menghapus `iam:CreateServiceLinkedRole` tindakan dari kebijakan. Jika Anda pernah membuat kluster Amazon EKS di akun Anda, maka peran ini sudah ada, kecuali Anda menghapusnya.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "eks:CreateCluster",
      "Resource": "arn:aws: eks:us-west-2:111122223333:cluster/my-cluster"
    },
    {
      "Effect": "Allow",
      "Action": "iam:CreateServiceLinkedRole",
      "Resource": "arn:aws: iam::111122223333:role/aws-service-role/eks.amazonaws.com/AWSServiceRoleForAmazonEKS",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "iam:AWSServiceName": "eks"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws: iam::111122223333:role/cluster-role-name"
    }
  ]
}
```

Buat kluster Kubernetes lokal di Outpost

Kebijakan contoh ini mencakup izin minimum yang diperlukan untuk membuat kluster lokal Amazon EKS yang diberi nama *my-cluster* di Pos Luar di *us-west-2* AWS Wilayah. Anda dapat mengganti AWS Region dengan AWS Region tempat Anda ingin membuat kluster. Jika Anda melihat peringatan yang mengatakan Tindakan dalam kebijakan Anda tidak mendukung izin tingkat sumber daya dan mengharuskan Anda untuk memilih AWS Management Console, tindakan tersebut dapat All resources diabaikan dengan aman. Jika akun Anda sudah memiliki AWSServiceRoleForAmazonEKSLocalOutpost peran, Anda dapat menghapus `iam:CreateServiceLinkedRole` tindakan dari kebijakan. Jika Anda pernah membuat kluster lokal Amazon EKS di Outpost di akun Anda, maka peran ini sudah ada, kecuali Anda menghapusnya.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "eks:CreateCluster",
      "Resource": "arn:aws: eks:us-west-2:111122223333:cluster/my-cluster"
    },
    {
      "Action": [
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcs",
        "iam:GetRole"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Effect": "Allow",
      "Action": "iam:CreateServiceLinkedRole",
      "Resource": "arn:aws: iam::111122223333:role/aws-service-role/outposts.eks-local.amazonaws.com/AWSServiceRoleForAmazonEKSLocalOutpost"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole",
        "iam:ListAttachedRolePolicies"
      ],
      "Resource": "arn:aws: iam::111122223333:role/cluster-role-name"
    }
  ]
}
```

```

    },
    {
      "Action": [
        "iam:CreateInstanceProfile",
        "iam:TagInstanceProfile",
        "iam:AddRoleToInstanceProfile",
        "iam:GetInstanceProfile",
        "iam>DeleteInstanceProfile",
        "iam:RemoveRoleFromInstanceProfile"
      ],
      "Resource": "arn:aws: iam::*:instance-profile/eks-local-*",
      "Effect": "Allow"
    },
  ],
}

```

Perbarui kluster Kubernetes

Kebijakan contoh ini mencakup izin minimum yang diperlukan untuk memperbarui kluster bernama *my-cluster* di Wilayah us-west-2 AWS .

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "eks:UpdateClusterVersion",
      "Resource": "arn:aws: eks:us-west-2:111122223333:cluster/my-cluster"
    }
  ]
}

```

Buat daftar atau deskripsikan semua kluster

Kebijakan contoh ini mencakup izin minimum yang diperlukan untuk mencantumkan dan menjelaskan semua kluster di akun Anda. [Prinsipal IAM](#) harus dapat membuat daftar dan mendeskripsikan cluster untuk menggunakan perintah CLI `update-kubeconfig` AWS .

```

{
  "Version": "2012-10-17",
  "Statement": [
    {

```

```
    "Effect": "Allow",
    "Action": [
        "eks:DescribeCluster",
        "eks:ListClusters"
    ],
    "Resource": "*"
}
]
```

Menggunakan peran tertaut layanan untuk Amazon EKS

[Amazon Elastic Kubernetes Service AWS menggunakan peran terkait layanan Identity and Access Management \(IAM\) and Access Management \(IAM\)](#). Peran tertaut layanan adalah jenis IAM role unik yang terkait langsung dengan Amazon EKS. Peran terkait layanan telah ditentukan sebelumnya oleh Amazon EKS dan mencakup semua izin yang diperlukan layanan untuk memanggil AWS layanan lain atas nama Anda.

Topik

- [Menggunakan peran untuk kluster Amazon EKS](#)
- [Menggunakan peran untuk grup simpul Amazon EKS](#)
- [Menggunakan peran untuk profil Amazon EKS Fargate](#)
- [Menggunakan peran untuk menghubungkan kluster Kubernetes ke Amazon EKS](#)
- [Menggunakan peran untuk kluster lokal Amazon EKS di Outpost](#)
- [Menggunakan peran untuk Dasbor Amazon EKS](#)

Menggunakan peran untuk kluster Amazon EKS

[Amazon Elastic Kubernetes Service AWS menggunakan peran terkait layanan Identity and Access Management \(IAM\) and Access Management \(IAM\)](#). Peran tertaut layanan adalah jenis IAM role unik yang terkait langsung dengan Amazon EKS. Peran terkait layanan telah ditentukan sebelumnya oleh Amazon EKS dan mencakup semua izin yang diperlukan layanan untuk memanggil AWS layanan lain atas nama Anda.

Peran terkait layanan membuat pengaturan Amazon EKS lebih mudah karena Anda tidak perlu menambahkan izin yang diperlukan secara manual. Amazon EKS menentukan izin atas peran tertaut layanan, dan jika tidak ada ketentuan lain, hanya Amazon EKS yang dapat menjalankan perannya.

Izin yang ditentukan mencakup kebijakan kepercayaan dan kebijakan izin, dan kebijakan izin tersebut tidak dapat dilampirkan ke entitas IAM lainnya.

Anda dapat menghapus peran terkait layanan hanya setelah menghapus sumber daya terkait terlebih dahulu. Ini melindungi sumber daya Amazon EKS Anda karena Anda tidak dapat secara tidak sengaja menghapus izin untuk mengakses sumber daya.

Untuk informasi tentang layanan lain yang mendukung peran terkait layanan, silakan lihat [layanan AWS yang bisa digunakan dengan IAM](#) dan carilah layanan yang memiliki opsi Ya di kolom Peran terkait layanan. Pilih Ya bersama tautan untuk melihat dokumentasi peran terkait layanan untuk layanan tersebut.

Izin peran terkait layanan untuk Amazon EKS

Amazon EKS menggunakan peran terkait layanan bernama `AWSServiceRoleForAmazonEKS`. Peran ini memungkinkan Amazon EKS untuk mengelola cluster di akun Anda. Kebijakan terlampir memungkinkan peran untuk mengelola sumber daya berikut: antarmuka jaringan, grup keamanan, log, dan VPCs file.

Note

Peran terkait layanan `AWSServiceRoleForAmazonEKS` berbeda dari peran yang diperlukan untuk pembuatan kluster. Untuk informasi selengkapnya, lihat [the section called “IAM role kluster”](#).

Peran terkait layanan `AWSServiceRoleForAmazonEKS` mempercayai layanan berikut untuk mengambil peran tersebut:

- `eks.amazonaws.com`

Kebijakan izin peran mengizinkan Amazon EKS untuk menyelesaikan tindakan berikut pada sumber daya yang ditentukan:

- [Amazon EKSService RolePolicy](#)

Anda harus mengonfigurasi izin untuk mengizinkan entitas IAM (seperti pengguna, grup, atau peran) untuk membuat, mengedit, atau menghapus peran terkait layanan. Untuk informasi selengkapnya, lihat [Izin peran terkait layanan](#) dalam Panduan Pengguna IAM.

Membuat peran tertaut layanan untuk Amazon EKS

Anda tidak perlu membuat peran terkait layanan secara manual. Saat Anda membuat klaster di AWS Management Console, AWS CLI, atau AWS API, Amazon EKS membuat peran terkait layanan untuk Anda.

Jika Anda menghapus peran tertaut layanan ini, dan ingin membuatnya lagi, Anda dapat mengulangi proses yang sama untuk membuat peran tersebut di akun Anda. Ketika Anda membuat klaster, Amazon EKS membuatkan peran tertaut layanan lagi untuk Anda.

Mengedit Peran tertaut layanan untuk Amazon EKS

Amazon EKS tidak mengizinkan Anda untuk mengedit peran tertaut layanan `AWSServiceRoleForAmazonEKS`. Setelah membuat peran tertaut layanan, Anda tidak dapat mengubah nama peran karena berbagai entitas mungkin mereferensikan peran tersebut. Namun, Anda dapat mengedit penjelasan peran menggunakan IAM. Untuk informasi selengkapnya, lihat [Mengedit peran tertaut layanan](#) dalam Panduan Pengguna IAM.

Menghapus peran tertaut layanan untuk Amazon EKS

Jika Anda tidak perlu lagi menggunakan fitur atau layanan yang memerlukan peran terkait layanan, kami merekomendasikan Anda menghapus peran tersebut. Dengan begitu, Anda tidak memiliki entitas yang tidak digunakan dan tidak dipantau atau dipelihara secara aktif. Namun, Anda harus membersihkan peran tertaut layanan terlebih dahulu sebelum dapat menghapusnya secara manual.

Membersihkan peran tertaut-layanan

Sebelum dapat menggunakan IAM untuk menghapus peran tertaut-layanan, Anda harus terlebih dahulu menghapus semua sumber daya yang digunakan oleh peran tersebut.

Note

Jika layanan Amazon EKS menggunakan peran saat Anda mencoba untuk menghapus sumber daya, maka penghapusan tersebut kemungkinan gagal. Jika hal itu terjadi, tunggu beberapa menit dan coba mengoperasikannya lagi.

1. Buka [konsol Amazon EKS](#).
2. Di panel navigasi sebelah kiri, pilih Klaster.

3. Jika klaster Anda memiliki grup simpul atau profil Fargate, Anda harus menghapusnya sebelum dapat menghapus klaster. Untuk informasi lebih lanjut, lihat [the section called “Hapus”](#) dan [the section called “Hapus profil”](#).
4. Pada halaman Klaster, pilih klaster yang ingin Anda hapus dan pilih Hapus.
5. Ketik nama klaster di jendela konfirmasi penghapusan, dan kemudian pilih Hapus.
6. Ulangi prosedur ini untuk klaster lain di akun Anda. Tunggu sampai semua operasi penghapusan selesai.

Hapus peran tertaut layanan secara manual

Gunakan konsol IAM, AWS CLI, atau AWS API untuk menghapus peran terkait layanan `AWSServiceRoleForAmazonEKS`. Untuk informasi selengkapnya, lihat [Menghapus peran tertaut layanan](#) dalam Panduan Pengguna IAM.

Wilayah yang didukung untuk peran tertaut-layanan Amazon EKS

Amazon EKS mendukung penggunaan peran tertaut-layanan di semua wilayah tempat layanan tersedia. Untuk informasi selengkapnya, lihat [titik akhir dan kuota Amazon EKS](#).

Menggunakan peran untuk grup simpul Amazon EKS

[Amazon EKS menggunakan peran AWS terkait layanan Identity and Access Management \(IAM\).](#)

Peran tertaut layanan adalah jenis IAM role unik yang terkait langsung dengan Amazon EKS. Peran terkait layanan telah ditentukan sebelumnya oleh Amazon EKS dan mencakup semua izin yang diperlukan layanan untuk memanggil AWS layanan lain atas nama Anda.

Peran terkait layanan membuat pengaturan Amazon EKS lebih mudah karena Anda tidak perlu menambahkan izin yang diperlukan secara manual. Amazon EKS menentukan izin atas peran tertaut layanan, dan jika tidak ada ketentuan lain, hanya Amazon EKS yang dapat menjalankan perannya. Izin yang ditentukan mencakup kebijakan kepercayaan dan kebijakan izin, dan kebijakan izin tersebut tidak dapat dilampirkan ke entitas IAM lainnya.

Anda dapat menghapus peran tertaut layanan hanya setelah menghapus sumber daya terkait terlebih dahulu. Ini melindungi sumber daya Amazon EKS Anda karena Anda tidak dapat secara tidak sengaja menghapus izin untuk mengakses sumber daya.

Untuk informasi tentang layanan lain yang mendukung peran terkait layanan, silakan lihat [layanan AWS yang bisa digunakan dengan IAM](#) dan carilah layanan yang memiliki opsi Ya di kolom Peran

terkait layanan. Pilih Ya bersama tautan untuk melihat dokumentasi peran tertaut layanan untuk layanan tersebut.

Izin peran tertaut layanan untuk Amazon EKS

Amazon EKS menggunakan peran terkait layanan bernama.

`AWSServiceRoleForAmazonEKSNodegroup` Peran ini memungkinkan Amazon EKS untuk mengelola grup node di akun Anda. `AWSServiceRoleForAmazonEKSNodegroupKebijakan` terlampir memungkinkan peran mengelola sumber daya berikut: Grup Auto Scaling, grup keamanan, templat peluncuran, dan profil instans IAM. Untuk informasi selengkapnya, lihat [the section called “AWS kebijakan terkelola: `AWSService RoleForAmazon EKSNodegroup`”](#).

Peran tertaut layanan `AWSServiceRoleForAmazonEKSNodegroup` mempercayai layanan berikut untuk mengambil peran tersebut:

- `eks-nodegroup.amazonaws.com`

Kebijakan izin peran mengizinkan Amazon EKS untuk menyelesaikan tindakan berikut pada sumber daya yang ditentukan:

- [AWSServiceRoleForAmazonEKSNodegroup](#)

Anda harus mengonfigurasi izin untuk mengizinkan entitas IAM (seperti pengguna, grup, atau peran) untuk membuat, mengedit, atau menghapus peran tertaut layanan. Untuk informasi selengkapnya, lihat [Izin peran tertaut layanan](#) dalam Panduan Pengguna IAM.

Membuat peran tertaut layanan untuk Amazon EKS

Anda tidak perlu membuat peran terkait layanan secara manual. Saat Anda `CreateNodegroup` berada di AWS Management Console, AWS CLI, atau AWS API, Amazon EKS membuat peran terkait layanan untuk Anda.

Important

Peran tertaut layanan ini dapat muncul di akun Anda jika Anda menyelesaikan tindakan di layanan lain yang menggunakan fitur yang disupport oleh peran ini. Jika Anda menggunakan layanan Amazon EKS sebelum 1 Januari 2017, ketika mulai mendukung peran terkait layanan, maka Amazon EKS membuat `AWSService RoleForAmazon EKSNodegroup` peran di akun Anda. Untuk mempelajari lebih lanjut, lihat [Peran baru muncul di akun IAM saya](#).

Membuat peran terkait layanan di Amazon EKS (API)AWS

Anda tidak perlu membuat peran terkait layanan secara manual. Saat Anda membuat grup node terkelola di AWS Management Console, AWS CLI, atau AWS API, Amazon EKS membuat peran terkait layanan untuk Anda.

Jika Anda menghapus peran tertaut layanan ini, dan ingin membuatnya lagi, Anda dapat mengulangi proses yang sama untuk membuat kembali peran tersebut di akun Anda. Ketika Anda membuat grup simpul terkelola lainnya, Amazon EKS membuatkan peran tertaut layanan lagi untuk Anda.

Mengedit peran tertaut-layanan untuk Amazon EKS

Amazon EKS tidak mengizinkan Anda untuk mengedit peran tertaut layanan `AWSServiceRoleForAmazonEKSCluster`. Setelah membuat peran tertaut layanan, Anda tidak dapat mengubah nama peran karena berbagai entitas mungkin mereferensikan peran tersebut. Namun, Anda dapat mengedit penjelasan peran menggunakan IAM. Untuk informasi selengkapnya, lihat [Mengedit peran tertaut layanan](#) dalam Panduan Pengguna IAM.

Menghapus peran tertaut layanan untuk Amazon EKS

Jika Anda tidak perlu lagi menggunakan fitur atau layanan yang memerlukan peran terkait layanan, kami merekomendasikan Anda menghapus peran tersebut. Dengan begitu, Anda tidak memiliki entitas yang tidak digunakan dan tidak dipantau atau dipelihara secara aktif. Namun, Anda harus membersihkan peran tertaut layanan terlebih dahulu sebelum dapat menghapusnya secara manual.

Membersihkan peran tertaut-layanan

Sebelum dapat menggunakan IAM untuk menghapus peran tertaut-layanan, Anda harus terlebih dahulu menghapus semua sumber daya yang digunakan oleh peran tersebut.

Note

Jika layanan Amazon EKS menggunakan peran saat Anda mencoba untuk menghapus sumber daya, maka penghapusan tersebut kemungkinan gagal. Jika hal itu terjadi, tunggu beberapa menit dan coba mengoperasikannya lagi.

1. Buka [konsol Amazon EKS](#).
2. Pada panel navigasi sebelah kiri, pilih Klaster.

3. Pilih tab Compute.
4. Di bagian Node groups, pilih grup node yang akan dihapus.
5. Ketik nama grup simpul di jendela konfirmasi penghapusan, dan kemudian pilih Hapus.
6. Ulangi prosedur ini untuk grup simpul lainnya dalam kluster. Tunggu sampai semua operasi penghapusan selesai.

Hapus peran tertaut layanan secara manual

Gunakan konsol IAM, AWS CLI, atau AWS API untuk menghapus peran terkait layanan `AWSServiceRoleForAmazonEKSNodegroup`. Untuk informasi selengkapnya, lihat [Menghapus peran tertaut layanan](#) dalam Panduan Pengguna IAM.

Wilayah yang didukung untuk peran tertaut-layanan Amazon EKS

Amazon EKS mendukung penggunaan peran tertaut-layanan di semua wilayah tempat layanan tersedia. Untuk informasi selengkapnya, lihat [titik akhir dan kuota Amazon EKS](#).

Menggunakan peran untuk profil Amazon EKS Fargate

[Amazon EKS menggunakan peran AWS terkait layanan Identity and Access Management \(IAM\).](#)

Peran tertaut layanan adalah jenis IAM role unik yang terkait langsung dengan Amazon EKS. Peran terkait layanan telah ditentukan sebelumnya oleh Amazon EKS dan mencakup semua izin yang diperlukan layanan untuk memanggil AWS layanan lain atas nama Anda.

Peran terkait layanan membuat pengaturan Amazon EKS lebih mudah karena Anda tidak perlu menambahkan izin yang diperlukan secara manual. Amazon EKS menentukan izin atas peran tertaut layanan, dan jika tidak ada ketentuan lain, hanya Amazon EKS yang dapat menjalankan perannya. Izin yang ditentukan mencakup kebijakan kepercayaan dan kebijakan izin, dan kebijakan izin tersebut tidak dapat dilampirkan ke entitas IAM lainnya.

Anda dapat menghapus peran tertaut layanan hanya setelah menghapus sumber daya terkait terlebih dahulu. Ini melindungi sumber daya Amazon EKS Anda karena Anda tidak dapat secara tidak sengaja menghapus izin untuk mengakses sumber daya.

Untuk informasi tentang layanan lain yang mendukung peran terkait layanan, silakan lihat [layanan AWS yang bisa digunakan dengan IAM](#) dan carilah layanan yang memiliki opsi Ya di kolom Peran terkait layanan. Pilih Ya bersama tautan untuk melihat dokumentasi peran tertaut layanan untuk layanan tersebut.

Izin peran tertaut layanan untuk Amazon EKS

Amazon EKS menggunakan peran terkait layanan bernama.

`AWSServiceRoleForAmazonEKSFargate` Peran ini memungkinkan Amazon EKS Fargate untuk mengonfigurasi jaringan VPC yang diperlukan untuk Pod Fargate. Kebijakan terlampir memungkinkan peran untuk membuat dan menghapus antarmuka jaringan elastis dan menggambarkan Antarmuka dan sumber daya jaringan elastis.

Peran tertaut layanan `AWSServiceRoleForAmazonEKSFargate` mempercayai layanan berikut untuk mengambil peran tersebut:

- `eks-fargate.amazonaws.com`

Kebijakan izin peran mengizinkan Amazon EKS untuk menyelesaikan tindakan berikut pada sumber daya yang ditentukan:

- [Amazon EKSFargateServiceRolePolicy](#)

Anda harus mengonfigurasi izin untuk mengizinkan entitas IAM (seperti pengguna, grup, atau peran) untuk membuat, mengedit, atau menghapus peran terkait layanan. Untuk informasi selengkapnya, lihat [Izin peran tertaut layanan](#) dalam Panduan Pengguna IAM.

Membuat peran tertaut layanan untuk Amazon EKS

Anda tidak perlu membuat peran terkait layanan secara manual. Saat Anda membuat profil Fargate di AWS Management Console, AWS CLI, atau API AWS, Amazon EKS membuat peran terkait layanan untuk Anda.

Important

Peran tertaut layanan ini dapat muncul di akun Anda jika Anda menyelesaikan tindakan di layanan lain yang menggunakan fitur yang disupport oleh peran ini. Jika Anda menggunakan layanan Amazon EKS sebelum 13 Desember 2019, ketika mulai mendukung peran terkait layanan, maka Amazon EKS membuat peran `AWSServiceRoleForAmazonEKSFargate` di akun Anda. Untuk mempelajari lebih lanjut, lihat [Peran baru muncul di akun IAM saya](#).

Membuat peran terkait layanan di Amazon EKS (API)AWS

Anda tidak perlu membuat peran terkait layanan secara manual. Saat Anda membuat profil Fargate di AWS Management Console, AWS CLI, atau API AWS, Amazon EKS membuat peran terkait layanan untuk Anda.

Jika Anda menghapus peran terkait layanan ini, dan ingin membuatnya lagi, Anda dapat mengulangi proses yang sama untuk membuat kembali peran tersebut di akun Anda. Ketika Anda membuat grup simpul terkelola lainnya, Amazon EKS membuatkan peran terkait layanan lagi untuk Anda.

Mengedit peran terkait-layanan untuk Amazon EKS

Amazon EKS tidak mengizinkan Anda untuk mengedit peran terkait layanan `AWSServiceRoleForAmazonEKSFargate`. Setelah membuat peran terkait layanan, Anda tidak dapat mengubah nama peran karena berbagai entitas mungkin mereferensikan peran tersebut. Namun, Anda dapat mengedit penjelasan peran menggunakan IAM. Untuk informasi selengkapnya, lihat [Mengedit peran terkait layanan](#) dalam Panduan Pengguna IAM.

Menghapus peran terkait layanan untuk Amazon EKS

Jika Anda tidak perlu lagi menggunakan fitur atau layanan yang memerlukan peran terkait layanan, kami merekomendasikan Anda menghapus peran tersebut. Dengan begitu, Anda tidak memiliki entitas yang tidak digunakan dan tidak dipantau atau dipelihara secara aktif. Namun, Anda harus membersihkan peran terkait layanan terlebih dahulu sebelum dapat menghapusnya secara manual.

Membersihkan peran terkait-layanan

Sebelum dapat menggunakan IAM untuk menghapus peran terkait-layanan, Anda harus terlebih dahulu menghapus semua sumber daya yang digunakan oleh peran tersebut.

Note

Jika layanan Amazon EKS menggunakan peran saat Anda mencoba untuk menghapus sumber daya, maka penghapusan tersebut kemungkinan gagal. Jika hal itu terjadi, tunggu beberapa menit dan coba mengoperasikannya lagi.

1. Buka [konsol Amazon EKS](#).
2. Pada panel navigasi sebelah kiri, pilih Klaster.

3. Pada halaman Klaster, pilih klaster Anda.
4. Pilih tab Compute.
5. Jika ada profil Fargate di bagian profil Fargate, pilih masing-masing satu per satu, lalu pilih Hapus.
6. Ketik nama profil di jendela konfirmasi penghapusan, dan kemudian pilih Hapus.
7. Ulangi prosedur ini untuk profil Fargate lainnya di dalam klaster dan untuk klaster lainnya di akun Anda.

Untuk menghapus peran tertaut-layanan secara manual

Gunakan konsol IAM, AWS CLI, atau API untuk menghapus AWS peran terkait layanan AWSService RoleForAmazon EKSTo Fargate. Untuk informasi selengkapnya, lihat [Menghapus peran tertaut layanan](#) dalam Panduan Pengguna IAM.

Wilayah yang didukung untuk peran tertaut-layanan Amazon EKS

Amazon EKS mendukung penggunaan peran tertaut-layanan di semua wilayah tempat layanan tersedia. Untuk informasi selengkapnya, lihat [titik akhir dan kuota Amazon EKS](#).

Menggunakan peran untuk menghubungkan klaster Kubernetes ke Amazon EKS

[Amazon EKS menggunakan peran AWS terkait layanan Identity and Access Management \(IAM\).](#)

Peran tertaut layanan adalah jenis IAM role unik yang terkait langsung dengan Amazon EKS. Peran terkait layanan telah ditentukan sebelumnya oleh Amazon EKS dan mencakup semua izin yang diperlukan layanan untuk memanggil AWS layanan lain atas nama Anda.

Peran terkait layanan membuat pengaturan Amazon EKS lebih mudah karena Anda tidak perlu menambahkan izin yang diperlukan secara manual. Amazon EKS menentukan izin atas peran tertaut layanan, dan jika tidak ada ketentuan lain, hanya Amazon EKS yang dapat menjalankan perannya. Izin yang ditentukan mencakup kebijakan kepercayaan dan kebijakan izin, dan kebijakan izin tersebut tidak dapat dilampirkan ke entitas IAM lainnya.

Anda dapat menghapus peran tertaut layanan hanya setelah menghapus sumber daya terkait terlebih dahulu. Ini melindungi sumber daya Amazon EKS Anda karena Anda tidak dapat secara tidak sengaja menghapus izin untuk mengakses sumber daya.

Untuk informasi tentang layanan lain yang mendukung peran terkait layanan, silakan lihat [layanan AWS yang bisa digunakan dengan IAM](#) dan carilah layanan yang memiliki opsi Ya di kolom Peran

terkait layanan. Pilih Ya bersama tautan untuk melihat dokumentasi peran tertaut layanan untuk layanan tersebut.

Izin peran tertaut layanan untuk Amazon EKS

Amazon EKS menggunakan peran terkait layanan bernama.

`AWSServiceRoleForAmazonEKSCloudConnector` Peran ini memungkinkan Amazon EKS untuk menghubungkan cluster Kubernetes. Kebijakan terlampir memungkinkan peran untuk mengelola sumber daya yang diperlukan untuk terhubung ke kluster Kubernetes Anda yang terdaftar.

Peran tertaut layanan `AWSServiceRoleForAmazonEKSCloudConnector` mempercayai layanan berikut untuk mengambil peran tersebut:

- `eks-connector.amazonaws.com`

Kebijakan izin peran mengizinkan Amazon EKS untuk menyelesaikan tindakan berikut pada sumber daya yang ditentukan:

- [Amazon EKSCloudConnector ServiceRolePolicy](#)

Anda harus mengonfigurasi izin untuk mengizinkan entitas IAM (seperti pengguna, grup, atau peran) untuk membuat, mengedit, atau menghapus peran terkait layanan. Untuk informasi selengkapnya, lihat [Izin peran tertaut layanan](#) dalam Panduan Pengguna IAM.

Membuat peran tertaut layanan untuk Amazon EKS

Anda tidak perlu membuat peran terkait layanan secara manual untuk menghubungkan kluster. Saat Anda menghubungkan cluster di AWS Management Console, AWS CLI, atau `AWS CLI eksctl`, Amazon EKS membuat peran terkait layanan untuk Anda.

Jika Anda menghapus peran tertaut layanan ini, dan ingin membuatnya lagi, Anda dapat mengulangi proses yang sama untuk membuat kembali peran tersebut di akun Anda. Saat Anda menghubungkan kluster, Amazon EKS membuat peran terkait layanan untuk Anda lagi.

Mengedit Peran tertaut layanan untuk Amazon EKS

Amazon EKS tidak mengizinkan Anda untuk mengedit peran tertaut layanan

`AWSServiceRoleForAmazonEKSCloudConnector`. Setelah membuat peran tertaut layanan, Anda tidak dapat mengubah nama peran karena berbagai entitas mungkin mereferensikan peran tersebut.

Namun, Anda dapat mengedit penjelasan peran menggunakan IAM. Untuk informasi selengkapnya, lihat [Mengedit peran tertaut layanan](#) dalam Panduan Pengguna IAM.

Menghapus peran tertaut layanan untuk Amazon EKS

Jika Anda tidak perlu lagi menggunakan fitur atau layanan yang memerlukan peran terkait layanan, kami merekomendasikan Anda menghapus peran tersebut. Dengan begitu, Anda tidak memiliki entitas yang tidak digunakan dan tidak dipantau atau dipelihara secara aktif. Namun, Anda harus membersihkan peran tertaut layanan terlebih dahulu sebelum dapat menghapusnya secara manual.

Membersihkan peran tertaut-layanan

Sebelum dapat menggunakan IAM untuk menghapus peran tertaut-layanan, Anda harus terlebih dahulu menghapus semua sumber daya yang digunakan oleh peran tersebut.

Note

Jika layanan Amazon EKS menggunakan peran saat Anda mencoba untuk menghapus sumber daya, maka penghapusan tersebut kemungkinan gagal. Jika hal itu terjadi, tunggu beberapa menit dan coba mengoperasikannya lagi.

1. Buka [konsol Amazon EKS](#).
2. Pada panel navigasi sebelah kiri, pilih Klaster.
3. Pada halaman Klaster, pilih klaster Anda.
4. Pilih tab Deregister dan kemudian pilih tab Ok.

Menghapus peran tertaut layanan secara manual

Gunakan konsol IAM, AWS CLI, atau AWS API untuk menghapus peran terkait layanan AWSService RoleForAmazonEKSCconnector . Untuk informasi selengkapnya, lihat [Menghapus peran tertaut layanan](#) dalam Panduan Pengguna IAM.

Menggunakan peran untuk kluster lokal Amazon EKS di Outpost

[Amazon Elastic Kubernetes Service AWS menggunakan peran terkait layanan Identity and Access Management \(IAM\) and Access Management \(IAM\)](#). Peran tertaut layanan adalah jenis IAM role unik yang terkait langsung dengan Amazon EKS. Peran terkait layanan telah ditentukan sebelumnya oleh

Amazon EKS dan mencakup semua izin yang diperlukan layanan untuk memanggil AWS layanan lain atas nama Anda.

Peran terkait layanan membuat pengaturan Amazon EKS lebih mudah karena Anda tidak perlu menambahkan izin yang diperlukan secara manual. Amazon EKS menentukan izin atas peran terkait layanan, dan jika tidak ada ketentuan lain, hanya Amazon EKS yang dapat menjalankan perannya. Izin yang ditentukan mencakup kebijakan kepercayaan dan kebijakan izin, dan kebijakan izin tersebut tidak dapat dilampirkan ke entitas IAM lainnya.

Anda dapat menghapus peran terkait layanan hanya setelah menghapus sumber daya terkait terlebih dahulu. Ini melindungi sumber daya Amazon EKS Anda karena Anda tidak dapat secara tidak sengaja menghapus izin untuk mengakses sumber daya.

Untuk informasi tentang layanan lain yang mendukung peran terkait layanan, silakan lihat [layanan AWS yang bisa digunakan dengan IAM](#) dan carilah layanan yang memiliki opsi Ya di kolom Peran terkait layanan. Pilih Ya bersama tautan untuk melihat dokumentasi peran terkait layanan untuk layanan tersebut.

Izin peran terkait layanan untuk Amazon EKS

Amazon EKS menggunakan peran terkait layanan bernama.

`AWSServiceRoleForAmazonEKSLocalOutpost` Peran ini memungkinkan Amazon EKS untuk mengelola cluster lokal di akun Anda. Kebijakan terlampir memungkinkan peran mengelola sumber daya berikut: antarmuka jaringan, grup keamanan, log, dan EC2 instans Amazon.

Note

Peran terkait layanan `AWSServiceRoleForAmazonEKSLocalOutpost` berbeda dari peran yang diperlukan untuk pembuatan kluster. Untuk informasi selengkapnya, lihat [the section called “IAM role kluster”](#).

Peran terkait layanan `AWSServiceRoleForAmazonEKSLocalOutpost` mempercayai layanan berikut untuk mengambil peran tersebut:

- `outposts.eks-local.amazonaws.com`

Kebijakan izin peran mengizinkan Amazon EKS untuk menyelesaikan tindakan berikut pada sumber daya yang ditentukan:

- [Amazon EKSService RolePolicy](#)

Anda harus mengonfigurasi izin untuk mengizinkan entitas IAM (seperti pengguna, grup, atau peran) untuk membuat, mengedit, atau menghapus peran terkait layanan. Untuk informasi selengkapnya, lihat [Izin peran tertaut layanan](#) dalam Panduan Pengguna IAM.

Membuat peran tertaut layanan untuk Amazon EKS

Anda tidak perlu membuat peran terkait layanan secara manual. Saat Anda membuat klaster di AWS Management Console, AWS CLI, atau AWS API, Amazon EKS membuat peran terkait layanan untuk Anda.

Jika Anda menghapus peran tertaut layanan ini, dan ingin membuatnya lagi, Anda dapat mengulangi proses yang sama untuk membuat peran tersebut di akun Anda. Ketika Anda membuat klaster, Amazon EKS membuatkan peran tertaut layanan lagi untuk Anda.

Mengedit Peran tertaut layanan untuk Amazon EKS

Amazon EKS tidak mengizinkan Anda untuk mengedit peran tertaut layanan `AWSServiceRoleForAmazonEKSLocalOutpost`. Setelah membuat peran terkait layanan, Anda tidak dapat mengubah nama peran karena berbagai entitas mungkin mereferensikan peran tersebut. Namun, Anda dapat mengedit penjelasan peran menggunakan IAM. Untuk informasi selengkapnya, lihat [Mengedit peran tertaut layanan](#) dalam Panduan Pengguna IAM.

Menghapus peran tertaut layanan untuk Amazon EKS

Jika Anda tidak perlu lagi menggunakan fitur atau layanan yang memerlukan peran terkait layanan, kami merekomendasikan Anda menghapus peran tersebut. Dengan begitu, Anda tidak memiliki entitas yang tidak digunakan dan tidak dipantau atau dipelihara secara aktif. Namun, Anda harus membersihkan peran tertaut layanan terlebih dahulu sebelum dapat menghapusnya secara manual.

Membersihkan peran tertaut-layanan

Sebelum dapat menggunakan IAM untuk menghapus peran tertaut-layanan, Anda harus terlebih dahulu menghapus semua sumber daya yang digunakan oleh peran tersebut.

 Note

Jika layanan Amazon EKS menggunakan peran saat Anda mencoba untuk menghapus sumber daya, maka penghapusan tersebut kemungkinan gagal. Jika hal itu terjadi, tunggu beberapa menit dan coba mengoperasikannya lagi.

1. Buka [konsol Amazon EKS](#).
2. Di panel navigasi kiri, pilih Amazon EKS Clusters.
3. Jika klaster Anda memiliki grup simpul atau profil Fargate, Anda harus menghapusnya sebelum dapat menghapus klaster. Untuk informasi lebih lanjut, lihat [the section called “Hapus”](#) dan [the section called “Hapus profil”](#).
4. Pada halaman Klaster, pilih klaster yang ingin Anda hapus dan pilih Hapus.
5. Ketik nama klaster di jendela konfirmasi penghapusan, dan kemudian pilih Hapus.
6. Ulangi prosedur ini untuk klaster lain di akun Anda. Tunggu sampai semua operasi penghapusan selesai.

Hapus peran tertaut layanan secara manual

Gunakan konsol IAM, AWS CLI, atau AWS API untuk menghapus peran terkait layanan `AWSServiceRoleForAmazonEKSLocalOutpost`. Untuk informasi selengkapnya, lihat [Menghapus peran tertaut layanan](#) dalam Panduan Pengguna IAM.

Wilayah yang didukung untuk peran tertaut-layanan Amazon EKS

Amazon EKS mendukung penggunaan peran tertaut-layanan di semua wilayah tempat layanan tersedia. Untuk informasi selengkapnya, lihat [titik akhir dan kuota Amazon EKS](#).

Menggunakan peran untuk Dasbor Amazon EKS

Dasbor Amazon EKS menggunakan peran terkait layanan ini untuk mengumpulkan informasi tentang sumber daya klaster dari beberapa wilayah dan akun. Dasbor terintegrasi dengan AWS Organizations untuk membaca informasi tentang beberapa akun di organisasi Anda dengan aman.

Untuk mempelajari lebih lanjut tentang Dasbor Amazon EKS, lihat [the section called “Dasbor Amazon EKS”](#).

Latar Belakang

[Amazon EKS menggunakan peran AWS terkait layanan Identity and Access Management \(IAM\).](#)

Peran terkait layanan adalah jenis IAM role unik yang terkait langsung dengan Amazon EKS. Peran terkait layanan telah ditentukan sebelumnya oleh Amazon EKS dan mencakup semua izin yang diperlukan layanan untuk memanggil AWS layanan lain atas nama Anda.

Peran terkait layanan membuat pengaturan Amazon EKS lebih mudah karena Anda tidak perlu menambahkan izin yang diperlukan secara manual. Amazon EKS menentukan izin atas peran terkait layanan, dan jika tidak ada ketentuan lain, hanya Amazon EKS yang dapat menjalankan perannya. Izin yang ditentukan mencakup kebijakan kepercayaan dan kebijakan izin, dan kebijakan izin tersebut tidak dapat dilampirkan ke entitas IAM lainnya.

Anda dapat menghapus peran terkait layanan hanya setelah menghapus sumber daya terkait terlebih dahulu. Ini melindungi sumber daya Amazon EKS Anda karena Anda tidak dapat secara tidak sengaja menghapus izin untuk mengakses sumber daya.

Untuk informasi tentang layanan lain yang mendukung peran terkait layanan, silakan lihat [layanan AWS yang bisa digunakan dengan IAM](#) dan carilah layanan yang memiliki opsi Ya di kolom Peran terkait layanan. Pilih Ya bersama tautan untuk melihat dokumentasi peran terkait layanan untuk layanan tersebut.

Izin peran terkait layanan untuk Amazon EKS

Amazon EKS menggunakan peran terkait layanan bernama.

`AWSServiceRoleForAmazonEKSDashboard` Peran ini memungkinkan Amazon EKS untuk menghasilkan dan menampilkan Dasbor EKS, termasuk informasi gabungan tentang sumber daya cluster. `AmazonEKSDashboardServiceRolePolicy` Kebijakan terlampir memungkinkan peran mengelola sumber daya berikut: Grup Auto Scaling, grup keamanan, templat peluncuran, dan profil instans IAM. Untuk informasi selengkapnya, lihat [the section called “AWS kebijakan terkelola: Amazon EKSDashboard ServiceRolePolicy”](#).

Peran terkait layanan `AWSServiceRoleForAmazonEKSDashboard` memercayai layanan berikut untuk mengambil peran tersebut:

- `dashboard.eks.amazonaws.com`

Untuk melihat versi terbaru dokumen kebijakan JSON, lihat [Amazon EKSDashboard ServiceRolePolicy](#) di Panduan Referensi Kebijakan AWS Terkelola.

Anda harus mengonfigurasi izin untuk mengizinkan entitas IAM (seperti pengguna, grup, atau peran) untuk membuat, mengedit, atau menghapus peran terkait layanan. Untuk informasi selengkapnya, lihat [Izin peran terkait layanan](#) dalam Panduan Pengguna IAM.

Membuat peran terkait layanan untuk Amazon EKS

Anda tidak perlu membuat peran terkait layanan secara manual. Saat Anda mengaktifkan dasbor di AWS konsol, Amazon EKS membuat peran terkait layanan untuk Anda.

Untuk informasi selengkapnya tentang mengaktifkan Dasbor EKS, lihat [the section called “Konfigurasi organisasi”](#).

Important

Peran terkait layanan ini dapat muncul di akun Anda jika Anda menyelesaikan tindakan di layanan lain yang menggunakan fitur yang disupport oleh peran ini.

Mengedit Peran terkait layanan untuk Amazon EKS

Amazon EKS tidak mengizinkan Anda untuk mengedit peran terkait layanan `AWSServiceRoleForAmazonEKSDashboard`. Setelah membuat peran terkait layanan, Anda tidak dapat mengubah nama peran karena berbagai entitas mungkin mereferensikan peran tersebut. Namun, Anda dapat mengedit penjelasan peran menggunakan IAM. Untuk informasi selengkapnya, lihat [Mengedit peran terkait layanan](#) dalam Panduan Pengguna IAM.

Menghapus peran terkait layanan untuk Amazon EKS

Jika Anda tidak perlu lagi menggunakan fitur atau layanan yang memerlukan peran terkait layanan, sebaiknya hapus peran tersebut. Dengan begitu, Anda tidak memiliki entitas yang tidak digunakan dan tidak dipantau atau dipelihara secara aktif. Namun, Anda harus membersihkan peran terkait layanan terlebih dahulu sebelum dapat menghapusnya secara manual.

Membersihkan peran terkait-layanan

Sebelum dapat menggunakan IAM untuk menghapus peran terkait-layanan, Anda harus terlebih dahulu menghapus semua sumber daya yang digunakan oleh peran tersebut.

Note

Jika layanan Amazon EKS menggunakan peran saat Anda mencoba untuk menghapus sumber daya, maka penghapusan tersebut kemungkinan gagal. Jika hal itu terjadi, tunggu beberapa menit dan coba mengoperasikannya lagi.

Untuk informasi selengkapnya tentang menonaktifkan Dasbor EKS, lihat [the section called “Konfigurasi organisasi”](#)

Menghapus peran tertaut layanan secara manual

Gunakan konsol IAM, AWS CLI, atau AWS API untuk menghapus peran terkait layanan `AWSServiceRoleForAmazonEKSDashboard`. Untuk informasi selengkapnya, lihat [Menghapus peran tertaut layanan](#) dalam Panduan Pengguna IAM.

Wilayah yang didukung untuk peran tertaut-layanan Amazon EKS

Amazon EKS mendukung penggunaan peran tertaut-layanan di semua wilayah tempat layanan tersedia. Untuk informasi selengkapnya, lihat [titik akhir dan kuota Amazon EKS](#).

Peran IAM Eksekusi Amazon EKS Pod

Peran eksekusi Amazon EKS Pod diperlukan untuk menjalankan Pod pada infrastruktur AWS Fargate.

Saat klaster Anda membuat Pod pada infrastruktur AWS Fargate, komponen yang berjalan pada infrastruktur Fargate harus melakukan panggilan atas nama Anda. AWS APIs ini agar mereka dapat melakukan tindakan seperti menarik gambar kontainer dari Amazon ECR atau merutekan log ke AWS layanan lain. Peran eksekusi Amazon EKS Pod memberikan izin IAM untuk melakukan ini.

Saat membuat profil Fargate, Anda harus menentukan peran eksekusi Pod untuk komponen Amazon EKS yang berjalan di infrastruktur Fargate menggunakan profil tersebut. Peran ini ditambahkan ke Kubernetes [Role based access control](#) (RBAC) klaster untuk otorisasi. Hal ini memungkinkan kubelet yang berjalan di infrastruktur Fargate untuk mendaftar dengan cluster Amazon EKS Anda sehingga dapat muncul di cluster Anda sebagai node.

Note

Profil Fargate harus memiliki peran IAM yang berbeda dari grup node Amazon EC2 .

⚠ Important

Container yang berjalan di Fargate Pod tidak dapat mengasumsikan izin IAM yang terkait dengan peran eksekusi Pod. Untuk memberikan izin pada container di Fargate Pod Anda untuk mengakses AWS layanan lain, Anda harus menggunakan [peran IAM](#) untuk akun layanan.

[Sebelum Anda membuat profil Fargate, Anda harus membuat peran IAM dengan Amazon. EKSFargate PodExecutionRolePolicy](#)

Periksa peran eksekusi Pod yang sudah dikonfigurasi dengan benar

Anda dapat menggunakan prosedur berikut untuk memeriksa dan melihat apakah akun Anda sudah memiliki peran eksekusi Amazon EKS Pod yang dikonfigurasi dengan benar. Untuk menghindari masalah keamanan wakil yang membingungkan, penting bahwa peran membatasi akses berdasarkan `SourceArn`. Anda dapat memodifikasi peran eksekusi sesuai kebutuhan untuk menyertakan dukungan untuk profil Fargate di cluster lain.

1. Buka konsol IAM di <https://console.aws.amazon.com/iam/>.
2. Di panel navigasi sebelah kiri, pilih Peran.
3. Pada halaman Peran, cari daftar peran untuk Amazon EKSFargate PodExecutionRole. Jika peran tidak ada, lihat [the section called “Membuat peran eksekusi Amazon EKS Pod”](#) untuk membuat peran. Jika peran itu memang ada, pilih perannya.
4. Di EKSFargate PodExecutionRole halaman Amazon, lakukan hal berikut:
 - a. Pilih Izin.
 - b. Pastikan kebijakan terkelola EKSFargatePodExecutionRolePolicyAmazon Amazon dilampirkan pada peran tersebut.
 - c. Pilih Hubungan kepercayaan.
 - d. Pilih Edit kebijakan kepercayaan.
5. Pada halaman Edit kebijakan kepercayaan, verifikasi bahwa hubungan kepercayaan berisi kebijakan berikut dan memiliki baris untuk profil Fargate di klaster Anda. Jika demikian, pilih Batalkan.

```
{  
  "Version": "2012-10-17",  
  "Statement": [  

```

```

{
  "Effect": "Allow",
  "Condition": {
    "ArnLike": {
      "aws:SourceArn": "arn:aws: eks:region-code:111122223333:fargateprofile/
my-cluster/*"
    }
  },
  "Principal": {
    "Service": "eks-fargate-pods.amazonaws.com"
  },
  "Action": "sts:AssumeRole"
}
]
}

```

Jika kebijakan cocok tetapi tidak memiliki baris yang menentukan profil Fargate di kluster, Anda dapat menambahkan baris berikut di bagian ArnLike atas objek. Ganti *region-code* dengan AWS Region tempat kluster Anda berada, *111122223333* dengan ID akun Anda, dan *my-cluster* dengan nama kluster Anda.

```
"aws:SourceArn": "arn:aws: eks:region-code:111122223333:fargateprofile/my-cluster/*",
```

Jika kebijakan tidak cocok, salin kebijakan lengkap sebelumnya ke dalam formulir dan pilih Perbarui kebijakan. Ganti *region-code* dengan AWS Wilayah tempat cluster Anda berada. Jika Anda ingin menggunakan peran yang sama di semua AWS Wilayah di akun Anda, ganti *region-code* dengan*. Ganti *111122223333* dengan ID akun Anda dan *my-cluster* dengan nama cluster Anda. Jika Anda ingin menggunakan peran yang sama untuk semua cluster di akun Anda, ganti *my-cluster* dengan*.

Membuat peran eksekusi Amazon EKS Pod

Jika Anda belum memiliki peran eksekusi Amazon EKS Pod untuk kluster Anda, Anda dapat menggunakan AWS Management Console atau AWS CLI untuk membuatnya.

AWS Management Console

- Buka konsol IAM di <https://console.aws.amazon.com/iam/>.
- Di panel navigasi sebelah kiri, pilih Peran.
- Pada halaman Peran, pilih Buat peran.

- d. Pada halaman Pilih entitas tepercaya, lakukan hal berikut:
 - i. Di bagian Jenis entitas tepercaya, pilih AWS layanan.
 - ii. Dari daftar dropdown Use case for other AWS services, pilih EKS.
 - iii. Pilih EKS - Fargate Pod.
 - iv. Pilih Berikutnya.
- e. Pada halaman Tambahkan izin, pilih Berikutnya.
- f. Pada halaman Nama, tinjau, dan buat, lakukan hal berikut:
 - i. Untuk nama Peran, masukkan nama unik untuk peran Anda, seperti `AmazonEKSFargatePodExecutionRole`.
 - ii. Di bawah Tambahkan tag (Opsional), tambahkan metadata ke peran dengan melampirkan tag sebagai pasangan nilai kunci. Untuk informasi selengkapnya tentang penggunaan tanda di IAM, lihat [Menandai sumber daya IAM](#) di Panduan Pengguna IAM.
 - iii. Pilih Buat peran.
- g. Pada halaman Peran, cari daftar peran untuk Amazon EKSFargate PodExecutionRole. Pilih peran.
- h. Di EKSFargate PodExecutionRole halaman Amazon, lakukan hal berikut:
 - i. Pilih Hubungan kepercayaan.
 - ii. Pilih Edit kebijakan kepercayaan.
- i. Pada halaman Edit kebijakan kepercayaan, lakukan hal berikut:
 - i. Salin dan tempel konten berikut ke dalam formulir Edit kebijakan kepercayaan. Ganti *region-code* dengan AWS Wilayah tempat cluster Anda berada. Jika Anda ingin menggunakan peran yang sama di semua AWS Wilayah di akun Anda, ganti *region-code* dengan `*`. Ganti `111122223333` dengan ID akun Anda dan *my-cluster* dengan nama cluster Anda. Jika Anda ingin menggunakan peran yang sama untuk semua cluster di akun Anda, ganti *my-cluster* dengan `*`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:eks:region-
code:111122223333:fargateprofile/my-cluster/*"
        }
      }
    }
  ]
}
```

```

    }
  },
  "Principal": {
    "Service": "eks-fargate-pods.amazonaws.com"
  },
  "Action": "sts:AssumeRole"
}
]
}

```

ii. Pilih Perbarui kebijakan.

AWS CLI

- a. Salin dan tempel konten berikut ke file bernama `pod-execution-role-trust-policy.json`. Ganti *region-code* dengan AWS Wilayah tempat cluster Anda berada. Jika Anda ingin menggunakan peran yang sama di semua AWS Wilayah di akun Anda, ganti *region-code* dengan `*`. Ganti `111122223333` dengan ID akun Anda dan *my-cluster* dengan nama cluster Anda. Jika Anda ingin menggunakan peran yang sama untuk semua cluster di akun Anda, ganti *my-cluster* dengan `*`.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:eks:region-code:111122223333:fargateprofile/my-cluster/*"
        }
      },
      "Principal": {
        "Service": "eks-fargate-pods.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}

```

- b. Buat peran IAM eksekusi Pod.

```

aws iam create-role \
  --role-name AmazonEKSFargatePodExecutionRole \

```

```
--assume-role-policy-document file://"pod-execution-role-trust-policy.json"
```

- c. Lampirkan kebijakan terkelola IAM Amazon EKS yang diperlukan untuk peran tersebut.

```
aws iam attach-role-policy \  
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSFargatePodExecutionRolePolicy \  
  --role-name AmazonEKSFargatePodExecutionRole
```

Peran IAM konektor Amazon EKS

Anda dapat menghubungkan kluster Kubernetes untuk melihatnya di kluster Anda. AWS Management Console Untuk terhubung ke kluster Kubernetes, buat peran IAM.

Periksa peran konektor EKS yang ada

Anda dapat menggunakan prosedur berikut untuk memeriksa dan melihat apakah akun Anda sudah memiliki peran konektor Amazon EKS.

1. Buka konsol IAM di <https://console.aws.amazon.com/iam/>.
2. Di panel navigasi sebelah kiri, pilih Peran.
3. Cari daftar peran untuk `AmazonEKSClusterAgentRole`. Jika peran yang menyertakan `AmazonEKSClusterAgentRole` tidak ada, maka lihat [the section called “Membuat peran agen konektor Amazon EKS”](#) untuk membuat peran tersebut. Jika peran yang mencakup `AmazonEKSClusterAgentRole` ada, kemudian pilih peran untuk melihat kebijakan terlampir.
4. Pilih Izin.
5. Pastikan kebijakan `EKSCluster AgentPolicy` terkelola Amazon melekat pada peran tersebut. Jika kebijakan dilampirkan, peran konektor Amazon EKS Anda dikonfigurasi dengan benar.
6. Pilih Trust relationship, lalu pilih Edit trust policy.
7. Verifikasi bahwa hubungan kepercayaan berisi kebijakan berikut. Jika hubungan kepercayaan sesuai dengan kebijakan berikut, pilih Cancel (Batalan). Jika hubungan kepercayaan tidak cocok, salin kebijakan ke jendela Edit kebijakan kepercayaan dan pilih Perbarui kebijakan.

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Principal": {
```

```

        "Service": [
            "ssm.amazonaws.com"
        ],
    },
    "Action": "sts:AssumeRole"
}
]
}

```

Membuat peran agen konektor Amazon EKS

Anda dapat menggunakan AWS Management Console atau AWS CloudFormation untuk membuat peran agen konektor.

AWS CLI

- a. Buat file bernama `eks-connector-agent-trust-policy.json` yang berisi JSON berikut untuk digunakan untuk peran IAM.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "ssm.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}

```

- b. Buat file bernama `eks-connector-agent-policy.json` yang berisi JSON berikut untuk digunakan untuk peran IAM.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "SsmControlChannel",

```

```

        "Effect": "Allow",
        "Action": [
            "ssmmessages:CreateControlChannel"
        ],
        "Resource": "arn:aws:eks:*:*:cluster/*"
    },
    {
        "Sid": "ssmDataplaneOperations",
        "Effect": "Allow",
        "Action": [
            "ssmmessages:CreateDataChannel",
            "ssmmessages:OpenDataChannel",
            "ssmmessages:OpenControlChannel"
        ],
        "Resource": "*"
    }
]
}

```

- c. Buat peran agen Amazon EKS Connector menggunakan kebijakan kepercayaan dan kebijakan yang Anda buat di item daftar sebelumnya.

```

aws iam create-role \
    --role-name AmazonEKSCoordinatorAgentRole \
    --assume-role-policy-document file://eks-coordinator-agent-trust-policy.json

```

- d. Lampirkan kebijakan ke peran agen Amazon EKS Connector Anda.

```

aws iam put-role-policy \
    --role-name AmazonEKSCoordinatorAgentRole \
    --policy-name AmazonEKSCoordinatorAgentPolicy \
    --policy-document file://eks-coordinator-agent-policy.json

```

AWS CloudFormation

- a. Simpan AWS CloudFormation template berikut ke file teks di sistem lokal Anda.

Note

Template ini juga menciptakan peran terkait layanan yang seharusnya dibuat saat `registerCluster` API dipanggil. Lihat [the section called “Peran konektor cluster”](#) untuk detail.

```
---
AWSTemplateFormatVersion: '2010-09-09'
Description: 'Provisions necessary resources needed to register clusters in EKS'
Parameters: {}
Resources:
  EKSConectorSLR:
    Type: AWS::IAM::ServiceLinkedRole
    Properties:
      AWSServiceName: eks-connector.amazonaws.com

  EKSConectorAgentRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Action: [ 'sts:AssumeRole' ]
            Principal:
              Service: 'ssm.amazonaws.com'

  EKSConectorAgentPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyName: EKSConectorAgentPolicy
      Roles:
        - {Ref: 'EKSConectorAgentRole'}
      PolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: 'Allow'
            Action: [ 'ssmmessages:CreateControlChannel' ]
            Resource:
              - Fn::Sub: 'arn:${AWS::Partition}:eks:*:*:cluster/*'
          - Effect: 'Allow'
            Action: [ 'ssmmessages:CreateDataChannel',
              'ssmmessages:OpenDataChannel', 'ssmmessages:OpenControlChannel' ]
            Resource: "*"

  Outputs:
    EKSConectorAgentRoleArn:
      Description: The agent role that EKS connector uses to communicate with AWS
        services.
```



```
Value: !GetAtt EKSCollectorAgentRole.Arn
```

- b. Buka [konsol AWS CloudFormation](#).
- c. Pilih Buat tumpukan dengan sumber daya baru (standar).
- d. Untuk Tentukan templat, pilih Unggah sebuah file templat, kemudian pilih Pilih file.
- e. Pilih file yang telah Anda buat sebelumnya, dan kemudian pilih Selanjutnya.
- f. Untuk Nama tumpukan, masukkan nama untuk peran Anda, misalnya `eksCollectorAgentRole`, kemudian pilih Selanjutnya.
- g. Pada halaman Konfigurasi opsi tumpukan, pilih Selanjutnya.
- h. Di halaman Tinjauan, tinjau informasi Anda, nyatakan bahwa tumpukan dapat membuat sumber daya IAM, kemudian pilih Buat.

AWS kebijakan terkelola untuk Amazon Elastic Kubernetes Service

Kebijakan AWS terkelola adalah kebijakan mandiri yang dibuat dan dikelola oleh AWS. AWS Kebijakan terkelola dirancang untuk memberikan izin bagi banyak kasus penggunaan umum sehingga Anda dapat mulai menetapkan izin kepada pengguna, grup, dan peran.

Perlu diingat bahwa kebijakan AWS terkelola mungkin tidak memberikan izin hak istimewa paling sedikit untuk kasus penggunaan spesifik Anda karena tersedia untuk digunakan semua pelanggan. AWS Kami menyarankan Anda untuk mengurangi izin lebih lanjut dengan menentukan [kebijakan yang dikelola pelanggan](#) yang khusus untuk kasus penggunaan Anda.

Anda tidak dapat mengubah izin yang ditentukan dalam kebijakan AWS terkelola. Jika AWS memperbarui izin yang ditentukan dalam kebijakan AWS terkelola, pemutakhiran akan memengaruhi semua identitas utama (pengguna, grup, dan peran) yang dilampirkan kebijakan tersebut. AWS kemungkinan besar akan memperbarui kebijakan AWS terkelola saat AWS layanan baru diluncurkan atau operasi API baru tersedia untuk layanan yang ada.

Untuk informasi selengkapnya, lihat [Kebijakan terkelola AWS](#) dalam Panduan Pengguna IAM.

AWS kebijakan terkelola: AmazonEKS_CNI_Policy

Anda dapat melampirkan `AmazonEKS_CNI_Policy` ke entitas IAM Anda. Sebelum Anda membuat grup EC2 node Amazon, kebijakan ini harus dilampirkan ke peran IAM [node, atau ke peran IAM](#) yang digunakan secara khusus oleh plugin Amazon VPC CNI untuk Kubernetes. Ini dimaksudkan agar dapat melakukan tindakan atas nama Anda. Kami menyarankan Anda melampirkan kebijakan

ke peran yang hanya digunakan oleh plugin. Untuk informasi selengkapnya, lihat [the section called “Amazon VPC CNI”](#) dan [the section called “Konfigurasi untuk IRSA”](#).

Detail izin

Kebijakan ini mencakup izin yang memungkinkan Amazon EKS untuk menyelesaikan tugas-tugas berikut:

- **ec2:*NetworkInterfaces** dan **ec2:*PrivateIpAddresses** — Memungkinkan plugin Amazon VPC CNI untuk melakukan tindakan seperti menyediakan Antarmuka Jaringan Elastis dan alamat IP untuk Pod untuk menyediakan jaringan untuk aplikasi yang berjalan di Amazon EKS.
- **ec2:DescribeInstances** dan **ec2:DescribeSubnets** — Memungkinkan plugin Amazon VPC CNI untuk melakukan tindakan seperti mendeskripsikan instance dan subnet untuk melihat jumlah alamat IP gratis di subnet VPC Amazon Anda. VPC CNI dapat menggunakan alamat IP gratis di setiap subnet untuk memilih subnet dengan alamat IP paling gratis untuk digunakan saat membuat elastic network interface.

Untuk melihat versi terbaru dari dokumen kebijakan JSON, lihat [AmazonEKS_CNI_Policy di Panduan Referensi Kebijakan](#) Terkelola. AWS

AWS kebijakan terkelola: EKSCluster Kebijakan Amazon

Anda dapat melampirkan `AmazonEKSClusterPolicy` ke entitas IAM Anda. Sebelum membuat kluster, Anda harus memiliki [IAM role kluster](#) dengan kebijakan terlampir ini. Cluster Kubernetes yang dikelola oleh Amazon EKS melakukan panggilan ke AWS layanan lain atas nama Anda. Mereka melakukan ini untuk mengelola sumber daya yang Anda gunakan dengan layanan.

Kebijakan ini mencakup izin yang memungkinkan Amazon EKS untuk menyelesaikan tugas-tugas berikut:

- **autoscaling:DescribeAutoScalingGroups** — Baca dan perbarui konfigurasi grup Auto Scaling. Izin ini tidak digunakan oleh Amazon EKS tetapi tetap dalam kebijakan untuk kompatibilitas mundur.
- **ec2:DescribeVolumes** dan **ec2:DescribeNetworkInterfaces** — Bekerja dengan volume dan sumber daya jaringan yang terkait dengan EC2 node Amazon. Hal ini diperlukan agar control plane Kubernetes dapat menggabungkan instance ke kluster dan secara dinamis menyediakan serta mengelola volume Amazon EBS yang diminta oleh volume persisten Kubernetes.
- **ec2:DeleteNetworkInterfaces** — Hapus antarmuka jaringan elastis yang dibuat oleh VPC CNI. Ini diperlukan agar EKS dapat membersihkan antarmuka jaringan elastis yang tertinggal jika VPC CNI berhenti secara tak terduga.

- **elasticloadbalancing**— Bekerja dengan Elastic Load Balancers dan tambahkan node ke dalamnya sebagai target. Ini diperlukan agar bidang kendali Kubernetes dapat secara dinamis menyediakan Penyeimbang Beban Elastis yang diminta oleh layanan Kubernetes.
- **iam**— Buat peran terkait layanan. Hal ini diperlukan agar control plane Kubernetes dapat secara dinamis menyediakan Elastic Load Balancer yang diminta oleh layanan Kubernetes.
- **kms**— Baca kunci dari AWS KMS. Ini diperlukan untuk pesawat kontrol Kubernetes untuk mendukung [enkripsi rahasia rahasia](#) Kubernetes yang disimpan di dalamnya. `etcd`

Untuk melihat versi terbaru dokumen kebijakan JSON, lihat Kebijakan [Amazon EKSCluster](#) di Panduan Referensi Kebijakan AWS Terkelola.

AWS kebijakan terkelola: Amazon EKSDashboard ConsoleReadOnly

Anda dapat melampirkan `AmazonEKSDashboardConsoleReadOnly` ke entitas IAM Anda.

Kebijakan ini mencakup izin yang memungkinkan Amazon EKS untuk menyelesaikan tugas-tugas berikut:

- **eks**- Akses hanya-baca ke data dasbor EKS, sumber daya, dan informasi versi cluster. Ini memungkinkan melihat metrik terkait EKS dan detail konfigurasi cluster.
- **organizations**- Akses read-only ke informasi AWS Organizations, termasuk:
 - Melihat detail organisasi dan akses layanan
 - Daftar akar organisasi, akun, dan unit organisasi
 - Melihat struktur organisasi

Untuk melihat versi terbaru dokumen kebijakan JSON, lihat [Amazon EKSDashboard ConsoleReadOnly](#) di Panduan Referensi Kebijakan AWS Terkelola.

AWS kebijakan terkelola: Amazon EKSFargate PodExecutionRolePolicy

Anda dapat melampirkan `AmazonEKSFargatePodExecutionRolePolicy` ke entitas IAM Anda. Sebelum Anda dapat membuat profil Fargate, Anda harus membuat peran eksekusi Fargate Pod dan melampirkan kebijakan ini padanya. Untuk informasi selengkapnya, lihat [the section called “Langkah 2: Buat peran eksekusi Fargate Pod”](#) dan [the section called “Tentukan profil”](#).

Kebijakan ini memberikan peran izin yang menyediakan akses ke sumber daya AWS layanan lain yang diperlukan untuk menjalankan Pod Amazon EKS di Fargate.

Detail izin

Kebijakan ini mencakup izin yang memungkinkan Amazon EKS untuk menyelesaikan tugas-tugas berikut:

- **ecr**— Memungkinkan Pod yang berjalan di Fargate untuk menarik gambar kontainer yang disimpan di Amazon ECR.

Untuk melihat versi terbaru dokumen kebijakan JSON, lihat [Amazon EKSFargate PodExecutionRolePolicy](#) di Panduan Referensi Kebijakan AWS Terkelola.

AWS kebijakan terkelola: Amazon EKSFargateServiceRolePolicy

Anda tidak dapat melampirkan AmazonEKSFargateServiceRolePolicy ke entitas IAM Anda. Kebijakan ini dilampirkan pada peran tertaut layanan yang mengizinkan Amazon EKS untuk melakukan tindakan atas nama Anda. Untuk informasi selengkapnya, lihat `AWSServiceRoleforAmazonEKSFargate`.

Kebijakan ini memberikan izin yang diperlukan kepada Amazon EKS untuk menjalankan tugas Fargate. Kebijakan ini hanya digunakan jika Anda memiliki simpul Fargate.

Detail izin

Kebijakan ini mencakup izin berikut yang memungkinkan Amazon EKS untuk menyelesaikan tugas berikut.

- **ec2**— Buat dan hapus Antarmuka Jaringan Elastis dan jelaskan Antarmuka dan sumber daya Jaringan Elastis. Ini diperlukan agar layanan Amazon EKS Fargate dapat mengonfigurasi jaringan VPC yang diperlukan untuk Pod Fargate.

Untuk melihat versi terbaru dokumen kebijakan JSON, lihat [Amazon EKSFargateServiceRolePolicy](#) di Panduan Referensi Kebijakan AWS Terkelola.

AWS kebijakan terkelola: EKSCoordinate Kebijakan Amazon

Anda dapat melampirkan AmazonEKSCoordinatePolicy ke entitas IAM Anda. Anda dapat melampirkan kebijakan ini ke [peran IAM kluster](#) Anda untuk memperluas sumber daya yang dapat dikelola EKS di akun Anda.

Kebijakan ini memberikan izin yang diperlukan Amazon EKS untuk membuat dan mengelola EC2 instance untuk kluster EKS, dan izin IAM yang diperlukan untuk konfigurasi. Selain itu, kebijakan ini memberikan izin kepada Amazon EKS untuk membuat peran terkait layanan EC2 Spot atas nama Anda.

Detail izin

Kebijakan ini mencakup izin yang memungkinkan Amazon EKS untuk menyelesaikan tugas-tugas berikut:

- **ec2**izin:
 - `ec2:CreateFleet` dan `ec2:RunInstances` - Memungkinkan membuat EC2 instance dan menggunakan EC2 sumber daya tertentu (gambar, grup keamanan, subnet) untuk node cluster EKS.
 - `ec2:CreateLaunchTemplate` - Memungkinkan membuat template EC2 peluncuran untuk node cluster EKS.
 - Kebijakan ini juga mencakup ketentuan untuk membatasi penggunaan EC2 izin ini ke sumber daya yang ditandai dengan nama kluster EKS dan tag lain yang relevan.
 - `ec2:CreateTags` - Memungkinkan menambahkan tag ke EC2 sumber daya yang dibuat oleh `CreateFleet`, `RunInstances`, dan `CreateLaunchTemplate` tindakan.
- **iam**izin:
 - `iam:AddRoleToInstanceProfile` - Memungkinkan menambahkan peran IAM ke profil instance komputasi EKS.
 - `iam:PassRole` - Memungkinkan meneruskan peran IAM yang diperlukan ke EC2 layanan.

Untuk melihat versi terbaru dokumen kebijakan JSON, lihat Kebijakan [Amazon EKSCompute](#) di Panduan Referensi Kebijakan AWS Terkelola.

AWS kebijakan terkelola: EKSNetworking Kebijakan Amazon

Anda dapat melampirkan `AmazonEKSNetworkingPolicy` ke entitas IAM Anda. Anda dapat melampirkan kebijakan ini ke [peran IAM kluster](#) Anda untuk memperluas sumber daya yang dapat dikelola EKS di akun Anda.

Kebijakan ini dirancang untuk memberikan izin yang diperlukan bagi Amazon EKS untuk membuat dan mengelola antarmuka jaringan untuk kluster EKS, memungkinkan bidang kontrol dan node pekerja untuk berkomunikasi dan berfungsi dengan baik.

Detail izin

Kebijakan ini memberikan izin berikut untuk mengizinkan Amazon EKS mengelola antarmuka jaringan untuk klaster:

- **ec2Izin Antarmuka Jaringan:**
 - `ec2:CreateNetworkInterface`- Memungkinkan membuat antarmuka EC2 jaringan.
 - Kebijakan ini mencakup ketentuan untuk membatasi penggunaan izin ini ke antarmuka jaringan yang ditandai dengan nama klaster EKS dan nama node CNI Kubernetes.
 - `ec2:CreateTags`- Memungkinkan menambahkan tag ke antarmuka jaringan yang dibuat oleh `CreateNetworkInterface` tindakan.
- **ec2Izin Manajemen Antarmuka Jaringan:**
 - `ec2:AttachNetworkInterface`, `ec2:DetachNetworkInterface` - Memungkinkan melampirkan dan melepaskan antarmuka jaringan ke instance. EC2
 - `ec2:UnassignPrivateIpAddresses`, `ec2:UnassignIpv6Addresses`, `ec2:AssignPrivateIpAddresses`, `ec2:AssignIpv6Addresses` - Memungkinkan mengelola penetapan alamat IP dari antarmuka jaringan.
 - Izin ini dibatasi untuk antarmuka jaringan yang ditandai dengan nama cluster EKS.

Untuk melihat versi terbaru dokumen kebijakan JSON, lihat Kebijakan [Amazon EKSNetworking](#) di Panduan Referensi Kebijakan AWS Terkelola.

AWS kebijakan terkelola: Amazon EKSBLOCK StoragePolicy

Anda dapat melampirkan `AmazonEKSBLOCKStoragePolicy` ke entitas IAM Anda. Anda dapat melampirkan kebijakan ini ke [peran IAM klaster](#) Anda untuk memperluas sumber daya yang dapat dikelola EKS di akun Anda.

Kebijakan ini memberikan izin yang diperlukan bagi Amazon EKS untuk membuat, mengelola, dan memelihara EC2 volume dan snapshot untuk klaster EKS, memungkinkan control plane dan node worker untuk menyediakan dan menggunakan penyimpanan persisten seperti yang dipersyaratkan oleh beban kerja Kubernetes.

Detail izin

Kebijakan IAM ini memberikan izin berikut untuk mengizinkan Amazon EKS mengelola EC2 volume dan snapshot:

- **ec2**Izin Manajemen Volume:
 - `ec2:AttachVolume,ec2:DetachVolume,ec2:ModifyVolume,ec2:EnableFastSnapshotRestores` - Memungkinkan melampirkan, melepaskan, memodifikasi, dan memungkinkan pemulihan snapshot cepat untuk volume. EC2
 - Izin ini dibatasi untuk volume yang ditandai dengan nama cluster EKS.
 - `ec2:CreateTags`- Memungkinkan menambahkan tag ke EC2 volume dan snapshot yang dibuat oleh `CreateVolume` dan `CreateSnapshot` tindakan.
- **ec2**Izin Pembuatan Volume:
 - `ec2:CreateVolume`- Memungkinkan membuat EC2 volume baru.
 - Kebijakan ini mencakup ketentuan untuk membatasi penggunaan izin ini ke volume yang ditandai dengan nama kluster EKS dan tag lain yang relevan.
 - `ec2:CreateSnapshot`- Memungkinkan membuat snapshot EC2 volume baru.
 - Kebijakan ini mencakup ketentuan untuk membatasi penggunaan izin ini pada snapshot yang ditandai dengan nama kluster EKS dan tag lain yang relevan.

Untuk melihat versi terbaru dokumen kebijakan JSON, lihat [Amazon EKSBLOCK StoragePolicy](#) di Panduan Referensi Kebijakan AWS Terkelola.

AWS kebijakan terkelola: Amazon EKSLoad BalancingPolicy

Anda dapat melampirkan `AmazonEKSLoadBalancingPolicy` ke entitas IAM Anda. Anda dapat melampirkan kebijakan ini ke [peran IAM kluster](#) Anda untuk memperluas sumber daya yang dapat dikelola EKS di akun Anda.

Kebijakan IAM ini memberikan izin yang diperlukan bagi Amazon EKS untuk bekerja dengan berbagai AWS layanan untuk mengelola Elastic Load Balancers () ELBs dan sumber daya terkait.

Detail izin

Izin utama yang diberikan oleh kebijakan ini adalah:

- **elasticloadbalancing**: Memungkinkan membuat, memodifikasi, dan mengelola Elastic Load Balancers dan Target Groups. Ini termasuk izin untuk membuat, memperbarui, dan menghapus penyeimbang beban, grup target, pendengar, dan aturan.
- **ec2**: Memungkinkan pembuatan dan pengelolaan grup keamanan, yang diperlukan untuk control plane Kubernetes untuk menggabungkan instance ke kluster dan mengelola volume Amazon EBS.

Juga memungkinkan mendeskripsikan dan mencantumkan EC2 sumber daya seperti instance,, Subnet VPCs, Grup Keamanan, dan sumber daya jaringan lainnya.

- **iam**: Memungkinkan pembuatan peran terkait layanan untuk Elastic Load Balancing, yang diperlukan agar bidang kontrol Kubernetes dapat disediakan secara dinamis. ELBs
- **kms**: Memungkinkan membaca kunci dari AWS KMS, yang diperlukan untuk control plane Kubernetes untuk mendukung enkripsi rahasia Kubernetes yang disimpan dalam etcd.
- **wafv2** dan **shield**: Memungkinkan mengaitkan dan memisahkan Web dan ACLs membuat/menghapus perlindungan AWS Shield untuk Elastic Load Balancer.
- **cognito-idp**, **acm**, and **elasticloadbalancing**: Memberikan izin untuk mendeskripsikan klien kumpulan pengguna, membuat daftar dan mendeskripsikan sertifikat, dan mendeskripsikan grup target, yang diperlukan untuk bidang kontrol Kubernetes untuk mengelola Elastic Load Balancers.

Kebijakan ini juga mencakup beberapa pemeriksaan kondisi untuk memastikan bahwa izin dicakup ke kluster EKS tertentu yang dikelola, menggunakan tag. `eks:eks-cluster-name`

Untuk melihat versi terbaru dokumen kebijakan JSON, lihat [Amazon EKSLoad BalancingPolicy](#) di Panduan Referensi Kebijakan AWS Terkelola.

AWS kebijakan terkelola: EKSService Kebijakan Amazon

Anda dapat melampirkan `AmazonEKSServicePolicy` ke entitas IAM Anda. Cluster yang dibuat sebelum 16 April 2020, mengharuskan Anda untuk membuat peran IAM dan melampirkan kebijakan ini padanya. Cluster yang dibuat pada atau setelah 16 April 2020, tidak mengharuskan Anda untuk membuat peran dan tidak mengharuskan Anda untuk menetapkan kebijakan ini. Saat Anda membuat kluster menggunakan prinsipal IAM yang memiliki `iam:CreateServiceLinkedRole` izin, peran terkait layanan [AWSServiceRoleforAmazonEKS](#) secara otomatis dibuat untuk Anda. Peran terkait layanan memiliki [kebijakan terkelola: Amazon EKSService RolePolicy](#) melekat padanya.

Kebijakan ini memungkinkan Amazon EKS untuk membuat dan mengelola sumber daya yang diperlukan guna mengoperasikan kluster Amazon EKS.

Detail izin

Kebijakan ini mencakup izin yang memungkinkan Amazon EKS untuk menyelesaikan tugas-tugas berikut.

- **eks**— Perbarui versi Kubernetes dari klaster Anda setelah Anda memulai pembaruan. Izin ini tidak digunakan oleh Amazon EKS tetapi tetap dalam kebijakan kompatibilitas mundur.
- **ec2**— Bekerja dengan Antarmuka Jaringan Elastis dan sumber daya dan tag jaringan lainnya. Ini diperlukan oleh Amazon EKS untuk mengonfigurasi jaringan yang memfasilitasi komunikasi antara simpul dan bidang kendali Kubernetes. Baca informasi tentang kelompok keamanan. Perbarui tag pada grup keamanan.
- **route53**— Kaitkan VPC dengan zona yang dihosting. Ini diperlukan oleh Amazon EKS untuk mengaktifkan jaringan titik akhir privat untuk server API klaster Kubernetes Anda.
- **logs**— Log peristiwa. Ini diperlukan agar Amazon EKS dapat mengirimkan log pesawat kontrol Kubernetes ke CloudWatch
- **iam**— Buat peran terkait layanan. Ini diperlukan agar Amazon EKS dapat membuat peran [the section called “Izin peran tertaut layanan untuk Amazon EKS”](#) terkait layanan atas nama Anda.

Untuk melihat versi terbaru dokumen kebijakan JSON, lihat Kebijakan [Amazon EKSService](#) di Panduan Referensi Kebijakan AWS Terkelola.

AWS kebijakan terkelola: Amazon EKSService RolePolicy

Anda tidak dapat melampirkan AmazonEKSServiceRolePolicy ke entitas IAM Anda. Kebijakan ini dilampirkan pada peran tertaut layanan yang mengizinkan Amazon EKS untuk melakukan tindakan atas nama Anda. Untuk informasi selengkapnya, lihat [the section called “Izin peran tertaut layanan untuk Amazon EKS”](#). Saat Anda membuat klaster menggunakan prinsipal IAM yang memiliki `iam:CreateServiceLinkedRole` izin, peran terkait layanan [AWSServiceRoleforAmazonEKS](#) secara otomatis dibuat untuk Anda dan kebijakan ini dilampirkan padanya.

Kebijakan ini memungkinkan peran terkait layanan untuk memanggil AWS layanan atas nama Anda.

Detail izin

Kebijakan ini mencakup izin berikut yang memungkinkan Amazon EKS untuk menyelesaikan tugas berikut.

- **ec2**— Buat dan jelaskan Antarmuka Jaringan Elastis dan EC2 instans Amazon, grup keamanan klaster, dan VPC yang diperlukan untuk membuat klaster. Untuk informasi selengkapnya, lihat [the section called “Persyaratan grup keamanan”](#). Baca informasi tentang kelompok keamanan. Perbarui tag pada grup keamanan. Baca informasi tentang Pemesanan Kapasitas Sesuai Permintaan.

- **ec2Mode Otomatis** - Hentikan EC2 instance yang dibuat oleh Mode Otomatis EKS. Untuk informasi selengkapnya, lihat [Mode Otomatis EKS](#).
- **iam**— Buat daftar semua kebijakan terkelola yang melekat pada peran IAM. Ini diperlukan agar Amazon EKS dapat mencantumkan dan memvalidasi semua kebijakan dan izin terkelola yang diperlukan untuk membuat kluster.
- Kaitkan VPC dengan zona yang dihosting — Ini diperlukan oleh Amazon EKS untuk mengaktifkan jaringan endpoint pribadi untuk server API cluster Kubernetes Anda.
- Peristiwa log - Ini diperlukan agar Amazon EKS dapat mengirimkan log pesawat kontrol Kubernetes ke. CloudWatch
- Masukkan metrik - Ini diperlukan agar Amazon EKS dapat mengirimkan log pesawat kontrol Kubernetes ke. CloudWatch
- **eks**- Kelola entri dan kebijakan akses kluster, memungkinkan kontrol halus atas siapa yang dapat mengakses sumber daya EKS dan tindakan apa yang dapat mereka lakukan. Ini termasuk mengaitkan kebijakan akses standar untuk operasi komputasi, jaringan, penyeimbangan beban, dan penyimpanan.
- **elasticloadbalancing**- Buat, kelola, dan hapus penyeimbang beban dan komponennya (pendengar, grup target, sertifikat) yang terkait dengan kluster EKS. Lihat atribut penyeimbang beban dan status kesehatan.
- **events**- Membuat dan mengelola EventBridge aturan untuk pemantauan EC2 dan peristiwa AWS Kesehatan yang terkait dengan kluster EKS, memungkinkan respons otomatis terhadap perubahan infrastruktur dan peringatan kesehatan.
- **iam**- Kelola profil EC2 instance dengan awalan “eks”, termasuk pembuatan, penghapusan, dan asosiasi peran, yang diperlukan untuk manajemen node EKS.
- **pricing& shield**- Akses informasi AWS harga dan status perlindungan Shield, memungkinkan manajemen biaya dan fitur keamanan canggih untuk sumber daya EKS.
- Pembersihan sumber daya - Hapus sumber daya yang ditandai EKS dengan aman termasuk volume, snapshot, template peluncuran, dan antarmuka jaringan selama operasi pembersihan kluster.

Untuk melihat versi terbaru dokumen kebijakan JSON, lihat [Amazon EKSService RolePolicy](#) di Panduan Referensi Kebijakan AWS Terkelola.

AWS kebijakan terkelola: EKSVPCResource Pengontrol Amazon

Anda dapat melampirkan kebijakan AmazonEKSVPCResourceController ke identitas IAM Anda. Jika Anda menggunakan [grup keamanan untuk Pod](#), Anda harus melampirkan kebijakan ini ke [peran IAM kluster Amazon EKS](#) untuk melakukan tindakan atas nama Anda.

Kebijakan ini memberikan izin peran kluster untuk mengelola Antarmuka Jaringan Elastis dan alamat IP untuk simpul.

Detail izin

Kebijakan ini mencakup izin yang memungkinkan Amazon EKS untuk menyelesaikan tugas-tugas berikut:

- **ec2**— Kelola Antarmuka Jaringan Elastis dan alamat IP untuk mendukung grup keamanan Pod dan node Windows.

Untuk melihat versi terbaru dokumen kebijakan JSON, lihat [EKSVPCResourcePengontrol Amazon](#) di Panduan Referensi Kebijakan AWS Terkelola.

AWS kebijakan terkelola: Amazon EKSWorker NodePolicy

Anda dapat melampirkan AmazonEKSWorkerNodePolicy ke entitas IAM Anda. Anda harus melampirkan kebijakan ini ke [peran IAM node](#) yang Anda tentukan saat membuat EC2 node Amazon yang memungkinkan Amazon EKS melakukan tindakan atas nama Anda. Jika Anda membuat grup simpul menggunakan eksctl, ia akan membuat IAM role simpul dan melampirkan kebijakan ini ke peran secara otomatis.

Kebijakan ini memberikan izin EC2 node Amazon EKS Amazon Amazon untuk terhubung ke kluster Amazon EKS.

Detail izin

Kebijakan ini mencakup izin yang memungkinkan Amazon EKS untuk menyelesaikan tugas-tugas berikut:

- **ec2**— Baca volume instance dan informasi jaringan. Hal ini diperlukan agar node Kubernetes dapat menjelaskan informasi tentang EC2 resource Amazon yang diperlukan node untuk bergabung dengan cluster Amazon EKS.

- **eks**— Secara opsional menggambarkan cluster sebagai bagian dari node bootstrapping.
- **eks-auth:AssumeRoleForPodIdentity**— Izinkan pengambilan kredensial untuk beban kerja EKS pada node. Hal ini diperlukan agar EKS Pod Identity berfungsi dengan baik.

Untuk melihat versi terbaru dokumen kebijakan JSON, lihat [Amazon EKSWorker NodePolicy](#) di Panduan Referensi Kebijakan AWS Terkelola.

AWS kebijakan terkelola: Amazon EKSWorker NodeMinimalPolicy

Anda dapat melampirkan Amazon EKSWorker NodeMinimalPolicy ke entitas IAM Anda. Anda dapat melampirkan kebijakan ini ke peran IAM node yang Anda tentukan saat membuat EC2 node Amazon yang memungkinkan Amazon EKS melakukan tindakan atas nama Anda.

Kebijakan ini memberikan izin EC2 node Amazon EKS Amazon Amazon untuk terhubung ke klaster Amazon EKS. Kebijakan ini memiliki izin yang lebih sedikit dibandingkan dengan Amazon EKSWorkerNodePolicy.

Detail izin

Kebijakan ini mencakup izin yang memungkinkan Amazon EKS untuk menyelesaikan tugas-tugas berikut:

- **eks-auth:AssumeRoleForPodIdentity**- Izinkan pengambilan kredensial untuk beban kerja EKS pada node. Hal ini diperlukan agar EKS Pod Identity berfungsi dengan baik.

Untuk melihat versi terbaru dokumen kebijakan JSON, lihat [Amazon EKSWorker NodePolicy](#) di Panduan Referensi Kebijakan AWS Terkelola.

AWS kebijakan terkelola: AWSService RoleForAmazon EKSNodegroup

Anda tidak dapat melampirkan AWSServiceRoleForAmazonEKSNodegroup ke entitas IAM Anda. Kebijakan ini dilampirkan pada peran tertaut layanan yang mengizinkan Amazon EKS untuk melakukan tindakan atas nama Anda. Untuk informasi selengkapnya, lihat [the section called “Izin peran tertaut layanan untuk Amazon EKS”](#).

Kebijakan ini memberikan izin AWSServiceRoleForAmazonEKSNodegroup peran yang memungkinkannya membuat dan mengelola grup EC2 node Amazon di akun Anda.

Detail izin

Kebijakan ini mencakup izin yang memungkinkan Amazon EKS untuk menyelesaikan tugas-tugas berikut:

- **ec2**— Bekerja dengan grup keamanan, tag, reservasi kapasitas, dan templat peluncuran. Ini diperlukan untuk grup node terkelola Amazon EKS untuk mengaktifkan konfigurasi akses jarak jauh dan untuk menjelaskan reservasi kapasitas yang dapat digunakan dalam grup node terkelola. Selain itu, grup simpul yang dikelola Amazon EKS membuat templat peluncuran atas nama Anda. Ini untuk mengonfigurasi grup EC2 Auto Scaling Amazon yang mendukung setiap grup node terkelola.
- **iam**— Buat peran terkait layanan dan berikan peran. Ini diperlukan oleh grup simpul yang dikelola Amazon EKS dalam mengelola profil instans untuk peran yang diteruskan ketika membuat grup simpul terkelola. Profil instance ini digunakan oleh EC2 instans Amazon yang diluncurkan sebagai bagian dari grup node terkelola. Amazon EKS perlu membuat peran terkait layanan untuk layanan lain seperti grup Amazon Auto EC2 Scaling. Izin ini digunakan dalam pembuatan grup node terkelola.
- **autoscaling**— Bekerja dengan grup Auto Scaling keamanan. Ini diperlukan oleh grup node terkelola Amazon EKS untuk mengelola grup EC2 Auto Scaling Amazon yang mendukung setiap grup node terkelola. Ini juga digunakan untuk mendukung fungsionalitas seperti mengusir Pod ketika node dihentikan atau didaur ulang selama pembaruan grup node.

Untuk melihat versi terbaru dari dokumen kebijakan JSON, lihat

[AWSServiceRoleForAmazonEKSNodegroup](#) di Panduan Referensi Kebijakan AWS Terkelola.

AWS kebijakan terkelola: Amazon EKSDashboard ServiceRolePolicy

Anda tidak dapat melampirkan `AmazonEKSDashboardServiceRolePolicy` ke entitas IAM Anda. Kebijakan ini dilampirkan pada peran tertaut layanan yang mengizinkan Amazon EKS untuk melakukan tindakan atas nama Anda. Untuk informasi selengkapnya, lihat [the section called “Izin peran tertaut layanan untuk Amazon EKS”](#).

Kebijakan ini memberikan izin `AWSServiceRoleForAmazonEKSDashboard` peran yang memungkinkannya membuat dan mengelola grup EC2 node Amazon di akun Anda.

Detail izin

Kebijakan ini mencakup izin berikut yang memungkinkan akses untuk menyelesaikan tugas-tugas ini:

- **organizations**— Lihat informasi tentang struktur dan akun AWS Organizations Anda. Ini termasuk izin untuk mencantumkan akun di organisasi Anda, melihat unit dan akar organisasi,

daftar administrator yang didelegasikan, melihat layanan yang memiliki akses ke organisasi Anda, dan mengambil informasi terperinci tentang organisasi dan akun Anda.

Untuk melihat versi terbaru dokumen kebijakan JSON, lihat [Amazon EKSDashboard ServiceRolePolicy](#) di Panduan Referensi Kebijakan AWS Terkelola.

AWS kebijakan terkelola: EBSCSIDriver Kebijakan Amazon

AmazonEBSCSIDriverPolicyKebijakan ini memungkinkan driver Amazon EBS Container Storage Interface (CSI) untuk membuat, memodifikasi, melampirkan, melepaskan, dan menghapus volume atas nama Anda. Ini termasuk memodifikasi tag pada volume yang ada dan mengaktifkan Pemulihan Snapshot Cepat (FSR) pada volume EBS. Ini juga memberikan izin driver EBS CSI untuk membuat, memulihkan, dan menghapus snapshot, dan untuk membuat daftar instance, volume, dan snapshot Anda.

Untuk melihat versi terbaru dokumen kebijakan JSON, lihat [Amazon EBSCSIDriver ServiceRolePolicy](#) di Panduan Referensi Kebijakan AWS Terkelola.

AWS kebijakan terkelola: EFSCSIDriver Kebijakan Amazon

AmazonEFSCSIDriverPolicyKebijakan ini memungkinkan Amazon EFS Container Storage Interface (CSI) untuk membuat dan menghapus titik akses atas nama Anda. Ini juga memberikan izin driver Amazon EFS CSI untuk mencantumkan sistem file titik akses Anda, target pemasangan, dan zona ketersediaan Amazon EC2 .

Untuk melihat versi terbaru dokumen kebijakan JSON, lihat [Amazon EFSCSIDriver ServiceRolePolicy](#) di Panduan Referensi Kebijakan AWS Terkelola.

AWS kebijakan terkelola: Amazon EKSLocal OutpostClusterPolicy

Anda dapat melampirkan kebijakan ini ke entitas IAM. Sebelum membuat kluster lokal, Anda harus melampirkan kebijakan ini ke [peran kluster](#) Anda. Cluster Kubernetes yang dikelola oleh Amazon EKS melakukan panggilan ke AWS layanan lain atas nama Anda. Mereka melakukan ini untuk mengelola sumber daya yang Anda gunakan dengan layanan.

AmazonEKSLocalOutpostClusterPolicyTermasuk izin berikut:

- **ec2tindakan baca** - Memungkinkan instance bidang kontrol untuk menggambarkan Availability Zone, tabel rute, instance, dan properti antarmuka jaringan. Izin yang diperlukan untuk EC2 instans Amazon agar berhasil bergabung dengan cluster sebagai instance bidang kontrol.

- **ssm**— Memungkinkan koneksi Amazon EC2 Systems Manager ke instance control plane, yang digunakan oleh Amazon EKS untuk berkomunikasi dan mengelola cluster lokal di akun Anda.
- **logs**— Memungkinkan instance untuk mendorong log ke Amazon CloudWatch.
- **secretsmanager**— Memungkinkan instance untuk mendapatkan dan menghapus data bootstrap untuk instance control plane dengan aman dari Secrets Manager AWS .
- **ecr**— Memungkinkan Pod dan kontainer yang berjalan pada instance control plane untuk menarik gambar kontainer yang disimpan di Amazon Elastic Container Registry.

Untuk melihat versi terbaru dokumen kebijakan JSON, lihat [Amazon EKSLocal OutpostClusterPolicy](#) di Panduan Referensi Kebijakan AWS Terkelola.

AWS kebijakan terkelola: Amazon EKSLocal OutpostServiceRolePolicy

Anda tidak dapat melampirkan kebijakan ini ke entitas IAM Anda. Saat Anda membuat kluster menggunakan prinsipal IAM yang memiliki `iam:CreateServiceLinkedRole` izin, Amazon EKS secara otomatis membuat peran terkait layanan [AWSServiceRoleforAmazonEKSLocalOutpost](#) untuk Anda dan melampirkan kebijakan ini padanya. Kebijakan ini memungkinkan peran terkait layanan untuk memanggil AWS layanan atas nama Anda untuk kluster lokal.

AmazonEKSLocalOutpostServiceRolePolicyTermasuk izin berikut:

- **ec2**— Memungkinkan Amazon EKS bekerja dengan keamanan, jaringan, dan sumber daya lainnya agar berhasil meluncurkan dan mengelola instans pesawat kontrol di akun Anda.
- **ssm, ssmmessages** — Memungkinkan koneksi Amazon EC2 Systems Manager ke instans bidang kontrol, yang digunakan oleh Amazon EKS untuk berkomunikasi dan mengelola kluster lokal di akun Anda.
- **iam**— Memungkinkan Amazon EKS mengelola profil instans yang terkait dengan instance bidang kontrol.
- **secretsmanager**- Memungkinkan Amazon EKS untuk menempatkan data bootstrap untuk instance control plane ke AWS Secrets Manager sehingga dapat direferensikan dengan aman selama bootstrap instance.
- **outposts**— Memungkinkan Amazon EKS mendapatkan informasi Outpost dari akun Anda agar berhasil meluncurkan cluster lokal di Outpost.

Untuk melihat versi terbaru dokumen kebijakan JSON, lihat [Amazon EKSLocal OutpostServiceRolePolicy](#) di Panduan Referensi Kebijakan AWS Terkelola.

Amazon EKS memperbarui kebijakan AWS terkelola

Lihat detail tentang pembaruan kebijakan AWS terkelola untuk Amazon EKS sejak layanan ini mulai melacak perubahan ini.

Untuk menerima pemberitahuan semua perubahan file sumber ke halaman dokumentasi khusus ini, Anda dapat berlangganan URL berikut dengan pembaca RSS:

<https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/security/iam-reference/security-iam-awsmanpol.adoc.atom>

Perubahan	Deskripsi	Tanggal
Menambahkan izin ke the section called “AWS kebijakan terkelola: Amazon EKSLocalOutpostServiceRolePolicy”	Menambahkan <code>ssmmessages:OpenDataChannel</code> izin ke <code>AmazonEKSLocalOutpostServiceRolePolicy</code> .	Juni 26, 2025
Menambahkan izin ke the section called “AWS kebijakan terkelola: EKSCompute Kebijakan Amazon” .	Izin sumber daya yang diperbarui untuk <code>ec2:CreateFleet</code> tindakan <code>ec2:RunInstances</code> dan untuk menyertakan reservasi <code>arn:aws:ec2:*:*:capacity-reservation/*</code> kapasitas. Hal ini memungkinkan Amazon EKS Auto Mode untuk meluncurkan instans dengan menggunakan Pemesanan Kapasitas EC2 Sesuai Permintaan di akun Anda. Ditambahkan <code>iam:CreateServiceLinkedRole</code> untuk memungkinkan Amazon EKS Auto Mode untuk membuat peran	Juni 20, 2025

Perubahan	Deskripsi	Tanggal
Menambahkan izin ke Amazon EKSService RolePolicy .	terkait layanan EC2 Spot AWSServiceRoleForEC2Spot atas nama Anda. Menambahkan ec2:DescribeCapacityReservations izin untuk mengizinkan Mode Otomatis Amazon EKS meluncurkan instans dengan menggunakan Reservasi Kapasitas EC2 Sesuai Permintaan di akun Anda.	Juni 20, 2025
Diperkenalkan the section called “AWS kebijakan terkelola: Amazon EKSDashboard ConsoleReadOnly” .	Memperkenalkan AmazonEKSDashboardConsoleReadOnly kebijakan baru.	Juni 19, 2025
Diperkenalkan the section called “AWS kebijakan terkelola: Amazon EKSDashboard ServiceRolePolicy” .	Memperkenalkan AmazonEKSDashboardServiceRolePolicy kebijakan baru.	21 Mei 2025
Menambahkan izin ke EKSClusterKebijakan Amazon .	Menambahkan ec2:DeleteNetworkInterfaces izin untuk mengizinkan Amazon EKS menghapus antarmuka jaringan elastis yang tertinggal jika VPC CNI berhenti secara tak terduga.	April 16, 2025

Perubahan	Deskripsi	Tanggal
Menambahkan izin ke Amazon EKSService RolePolicy .	Ditambahkan <code>ec2:RevokeSecurityGroupEgress</code> dan <code>ec2:AuthorizeSecurityGroupEgress</code> izin untuk memungkinkan AI/ML pelanggan EKS menambahkan aturan Security Group Egress ke EKS Cluster SG default yang kompatibel dengan EFA, sebagai bagian dari rilis versi EKS 1.33.	April 14, 2025
Menambahkan izin ke Amazon EKSService RolePolicy .	Menambahkan izin untuk menghentikan EC2 instance yang dibuat oleh EKS Auto Mode.	Februari 28, 2025

Perubahan	Deskripsi	Tanggal
Menambahkan izin ke EBSCSIDriverKebijakan Amazon .	Menambahkan pernyataan baru yang mengotorisasi Driver EBS CSI untuk memulihkan semua snapshot. Ini sebelumnya diizinkan oleh kebijakan yang ada tetapi pernyataan eksplisit baru diperlukan karena perubahan dalam penanganan IAM untuk <code>CreateVolume</code>. Ditambahkan kemampuan untuk EBS CSI Driver untuk memodifikasi tag pada volume yang ada. Driver EBS CSI dapat memodifikasi tag volume yang ada melalui parameter di Kubernetes. <code>VolumeAttributesClasses</code> Menambahkan kemampuan untuk EBS CSI Driver untuk mengaktifkan Fast Snapshot Restore (FSR) pada volume EBS. Driver EBS CSI dapat mengaktifkan FSR pada volume baru melalui parameter di kelas penyimpanan Kubernetes.	Januari 13, 2025
Menambahkan izin ke the section called “AWS kebijakan terkelola: Amazon EKSLoad BalancingPolicy” .	Diperbarui AmazonEKS <code>LoadBalancingPolicy</code> untuk memungkinkan daftar dan mendeskripsikan sumber daya jaringan dan alamat IP.	Desember 26, 2024

Perubahan	Deskripsi	Tanggal
Menambahkan izin ke the section called “AWS kebijakan terkelola: AWSServiceRoleForAmazonEKSNodegroup” .	Diperbarui AWSServiceRoleForAmazonEKSNodegroup untuk kompatibilitas dengan wilayah Tiongkok.	November 22, 2024
Menambahkan izin ke the section called “AWS kebijakan terkelola: Amazon EKSLocalOutpostClusterPolicy”	Menambahkan ec2:DescribeAvailabilityZones izin AmazonEKSLocalOutpostClusterPolicy agar AWS Cloud Controller Manager pada bidang kontrol cluster dapat mengidentifikasi Availability Zone tempat setiap node berada.	November 21, 2024
Menambahkan izin ke the section called “AWS kebijakan terkelola: AWSServiceRoleForAmazonEKSNodegroup” .	AWSServiceRoleForAmazonEKSNodegroup Kebijakan yang diperbarui ec2:RebootInstances untuk mengizinkan instance yang dibuat oleh grup node terkelola Amazon EKS. Membatasi ec2:CreateTags izin untuk EC2 sumber daya Amazon.	November 20, 2024

Perubahan	Deskripsi	Tanggal
Menambahkan izin ke the section called “AWS kebijakan terkelola: Amazon EKSServiceRolePolicy” .	EKS memperbarui kebijakan AWS terkelolaAmazonEKSServiceRolePolicy . Menambahkan izin untuk kebijakan akses EKS, manajemen penyeimbang beban, dan pembersihan sumber daya klaster otomatis.	November 16, 2024
Diperkenalkan the section called “AWS kebijakan terkelola: EKSCompute Kebijakan Amazon” .	EKS memperbarui kebijakan AWS terkelolaAmazonEKSComputePolicy . Izin sumber daya yang diperbarui untuk iam:AddRoleToInstanceProfile tindakan tersebut.	November 7, 2024
Diperkenalkan the section called “AWS kebijakan terkelola: EKSCompute Kebijakan Amazon” .	AWS memperkenalkanAmazonEKSComputePolicy .	November 1, 2024
Menambahkan izin ke AmazonEKSClusterPolicy	Menambahkan ec2:DescribeInstanceTopology izin untuk mengizinkan Amazon EKS melampirkan informasi topologi ke node sebagai label.	November 1, 2024
Diperkenalkan the section called “AWS kebijakan terkelola: Amazon EKSBlockStoragePolicy” .	AWS memperkenalkanAmazonEKSBlockStoragePolicy .	Oktober 30, 2024

Perubahan	Deskripsi	Tanggal
Diperkenalkan the section called “AWS kebijakan terkelola: Amazon EKSLoad BalancingPolicy” .	AWS memperkenalkan AmazonEKSLoadBalancingPolicy .	Oktober 30, 2024
Menambahkan izin ke Amazon EKSService RolePolicy .	Menambahkan cloudwatch:PutMetricData izin untuk mengizinkan Amazon EKS mempublikasikan metrik ke Amazon. CloudWatch	Oktober 29, 2024
Diperkenalkan the section called “AWS kebijakan terkelola: EKSNetworking Kebijakan Amazon” .	AWS memperkenalkan AmazonEKSNetworkingPolicy .	Oktober 28, 2024
Menambahkan izin ke dan AmazonEKSServicePolicy AmazonEKSServiceRolePolicy	Izin tag yang ditambahkan ec2:GetSecurityGroupsForVpc dan terkait untuk memungkinkan EKS membaca informasi grup keamanan dan memperbarui tag terkait.	Oktober 10, 2024
Memperkenalkan Amazon EKSWorker NodeMinimalPolicy .	AWS memperkenalkan AmazonEKSWorkerNodeMinimalPolicy .	Oktober 3, 2024

Perubahan	Deskripsi	Tanggal
Menambahkan izin ke AWSServiceRoleForAmazonEKSNodegroup .	Ditambahkan <code>autoscaling:ResumeProcesses</code> dan <code>autoscaling:SuspendProcesses</code> izin untuk memungkinkan Amazon EKS menangguhkan dan melanjutkan di grup Auto Scaling yang dikelola <code>AZRebalance</code> Amazon EKS.	Agustus 21, 2024
Menambahkan izin ke AWSServiceRoleForAmazonEKSNodegroup .	Menambahkan <code>ec2:DescribeCapacityReservations</code> izin untuk memungkinkan Amazon EKS menjelaskan reservasi kapasitas di akun pengguna. Menambahkan <code>autoscaling:PutScheduledUpdateGroupAction</code> izin untuk mengaktifkan pengaturan penskalaan terjadwal pada grup <code>CAPACITY_BLOCK</code> node.	27 Juni 2024

Perubahan	Deskripsi	Tanggal
AmazonEKS_CNI_Policy - Update ke kebijakan yang ada	Amazon EKS menambahkan <code>ec2:DescribeSubnets</code> izin baru untuk memungkinkan plugin Amazon VPC CNI untuk Kubernetes melihat jumlah alamat IP gratis di subnet VPC Amazon Anda. VPC CNI dapat menggunakan alamat IP gratis di setiap subnet untuk memilih subnet dengan alamat IP paling gratis untuk digunakan saat membuat elastic network interface.	Maret 4, 2024
Amazon EKSWorker NodePolicy - Perbarui ke kebijakan yang ada	Amazon EKS menambahkan izin baru untuk mengizinkan Identitas Pod EKS. Amazon EKS Pod Identity Agent menggunakan peran node.	26 November 2023
Memperkenalkan EFSCSIDriverKebijakan Amazon .	AWS memperkenalkan <code>AmazonEFSCSIDriverPolicy</code> .	26 Juli 2023
Menambahkan izin ke EKSClusterKebijakan Amazon .	Menambahkan <code>ec2:DescribeAvailabilityZones</code> izin untuk mengizinkan Amazon EKS mendapatkan detail AZ selama penemuan otomatis subnet sambil membuat penyeimbang beban.	7 Februari 2023

Perubahan	Deskripsi	Tanggal
Ketentuan kebijakan yang diperbarui di EBSCSIDriverKebijakan Amazon .	Menghapus kondisi kebijakan yang tidak valid dengan karakter wildcard di bidang kunci. StringLike Juga menambahkan kondisi <code>ec2:ResourceTag/kuernetes.io/created-for/pvc/name: "*" baru</code> <code>ec2:DeleteVolume</code> , yang memungkinkan driver EBS CSI untuk menghapus volume yang dibuat oleh plugin in-tree.	17 November 2022
Menambahkan izin ke Amazon EKSLocal OutpostServiceRole Policy .	Ditambahkan <code>ec2:DescribeVPCAttribute</code> , <code>ec2:GetConsoleOutput</code> dan <code>ec2:DescribeSecret</code> untuk memungkinkan validasi prasyarat yang lebih baik dan kontrol siklus hidup terkelola. Juga ditambahkan <code>ec2:DescribePlacementGroups</code> dan <code>"arn:aws: ec2:*:*:placement-group/*" ec2:RunInstances</code> untuk mendukung kontrol penempatan pesawat kontrol EC2 instans Amazon di Outposts.	24 Oktober 2022

Perubahan	Deskripsi	Tanggal
Perbarui izin Amazon Elastic Container Registry di Amazon EKSLocal OutpostClusterPolicy .	Tindakan yang dipindahkan <code>ecr:GetDownloadUrlForLayer</code> dari semua bagian sumber daya ke bagian cakupan. Menambahkan sumber daya <code>arn:aws:ecr:*:*:repository/eks/</code> . Sumber daya yang dihapus <code>arn:aws:ecr:.</code> . Sumber daya ini dicakup oleh <code>arn:aws:ecr:*:*:repository/eks/*</code> sumber daya tambahan.	20 Oktober 2022
Menambahkan izin ke Amazon EKSLocal OutpostClusterPolicy .	Menambahkan repositori <code>arn:aws:ecr:*:*:repository/kubelet-config-updater</code> Amazon Elastic Container Registry sehingga instance bidang kontrol cluster dapat memperbarui beberapa argumen. <code>kubelet</code>	31 Agustus 2022
Memperkenalkan Amazon EKSLocal OutpostClusterPolicy .	AWS memperkenalkan <code>AmazonEKSLocalOutpostClusterPolicy</code> .	Agustus 24, 2022
Memperkenalkan Amazon EKSLocal OutpostServiceRolePolicy .	AWS memperkenalkan <code>AmazonEKSLocalOutpostServiceRolePolicy</code> .	23 Agustus 2022

Perubahan	Deskripsi	Tanggal
Memperkenalkan EBSCSIDriverPolicy kebijakan Amazon.	AWS memperkenalkan AmazonEBS CSIDriverPolicy .	4 April 2022
Menambahkan izin ke AmazonEKSTaskRolePolicy .	Ditambahkan <code>ec2:DescribeInstanceTypes</code> untuk mengaktifkan Amazon EKS yang dioptimalkan AMIs yang dapat menemukan properti tingkat instans secara otomatis.	Maret 21, 2022
Menambahkan izin ke AWSServiceRoleForAmazonEKSTaskRolePolicy .	Menambahkan <code>autoscaling:EnableMetricsCollection</code> izin untuk mengizinkan Amazon EKS mengaktifkan pengumpulan metrik.	13 Desember 2021
Menambahkan izin ke EKSClusterKebijakan Amazon .	Izin <code>ec2:DescribeAccountAttributes</code> , <code>ec2:DescribeAddresses</code> , dan <code>ec2:DescribeInternetGateways</code> ditambahkan agar Amazon EKS diizinkan untuk membuat peran tertaut layanan bagi Penyeimbang Beban Jaringan.	17 Juni 2021
Amazon EKS mulai melacak perubahan.	Amazon EKS mulai melacak perubahan untuk kebijakan yang AWS dikelola.	17 Juni 2021

Menyelesaikan masalah IAM

Topik ini mencakup beberapa kesalahan umum yang mungkin Anda temui saat menggunakan Amazon EKS dengan IAM dan cara yang dilakukan untuk mengatasinya.

AccessDeniedException

Jika Anda menerima `AccessDeniedException` saat memanggil operasi AWS API, kredensial [utama IAM](#) yang Anda gunakan tidak memiliki izin yang diperlukan untuk melakukan panggilan itu.

```
An error occurred (AccessDeniedException) when calling the DescribeCluster operation:
User: arn:aws:iam::111122223333:user/user_name is not authorized to perform:
eks:DescribeCluster on resource: arn:aws:eks:region:111122223333:cluster/my-cluster
```

Dalam pesan contoh sebelumnya, pengguna tidak memiliki izin untuk memanggil operasi Amazon EKS `DescribeCluster` API. Untuk memberikan izin admin Amazon EKS ke kepala sekolah IAM, lihat [the section called “Kebijakan berbasis identitas”](#)

Untuk informasi umum selengkapnya tentang IAM, tinjau [Mengontrol akses menggunakan kebijakan](#) di Panduan Pengguna IAM.

Tidak dapat melihat Node di tab Compute atau apa pun di tab Resources dan Anda menerima kesalahan di AWS Management Console

Anda mungkin melihat pesan kesalahan konsol yang menyatakan `Your current user or role does not have access to Kubernetes objects on this EKS cluster`. Pastikan bahwa pengguna [utama IAM](#) yang Anda gunakan AWS Management Console dengan memiliki izin yang diperlukan. Untuk informasi selengkapnya, lihat [the section called “Izin yang diperlukan”](#).

aws-auth **ConfigMap** tidak memberikan akses ke cluster

[AWS IAM Authenticator](#) tidak mengizinkan jalur dalam peran ARN yang digunakan dalam.

ConfigMap Karena itu, sebelum Anda tentukan `roleARN`, hapus jalur. Misalnya, perubahan `arn:aws:iam::111122223333:role/team/developers/eks-admin` ke `arn:aws:iam::111122223333:role/eks-admin`.

Saya tidak berwenang untuk melakukan `iam:PassRole`

Jika Anda menerima kesalahan bahwa Anda tidak diizinkan untuk melakukan `iam:PassRole` tindakan, kebijakan Anda harus diperbarui agar Anda dapat meneruskan peran ke Amazon EKS.

Beberapa AWS layanan memungkinkan Anda untuk meneruskan peran yang ada ke layanan tersebut alih-alih membuat peran layanan baru atau peran terkait layanan. Untuk melakukannya, Anda harus memiliki izin untuk mendapatkan peran ke layanan.

Contoh kesalahan berikut terjadi saat pengguna IAM bernama `marymajor` mencoba menggunakan konsol tersebut untuk performa tindakan di Amazon EKS. Namun, tindakan tersebut memerlukan layanan untuk mendapatkan izin yang diberikan oleh peran layanan. Mary tidak memiliki izin untuk meneruskan peran tersebut pada layanan.

```
User: {arn-aws}iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

Dalam hal ini, kebijakan Mary harus diperbarui untuk memungkinkannya melakukan `iam:PassRole` tindakan.

Jika Anda memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Saya ingin mengizinkan orang di luar AWS akun saya untuk mengakses sumber daya Amazon EKS saya

Anda dapat membuat peran yang dapat digunakan pengguna di akun lain atau orang-orang di luar organisasi Anda untuk mengakses sumber daya Anda. Anda dapat menentukan siapa saja yang dipercaya untuk mengambil peran tersebut. Untuk layanan yang mendukung kebijakan berbasis sumber daya atau daftar kontrol akses (ACLs), Anda dapat menggunakan kebijakan tersebut untuk memberi orang akses ke sumber daya Anda.

Untuk mempelajari selengkapnya, konsultasikan hal berikut:

- Untuk mempelajari apakah Amazon EKS mendukung fitur ini, lihat [the section called “Amazon EKS dan IAM”](#).
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda di seluruh AWS akun yang Anda miliki, lihat [Menyediakan akses ke pengguna IAM di AWS akun lain yang Anda miliki](#) di Panduan Pengguna IAM.
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda ke AWS akun pihak ketiga, lihat [Menyediakan akses ke AWS akun yang dimiliki oleh pihak ketiga](#) di Panduan Pengguna IAM.
- Untuk mempelajari cara memberikan akses melalui federasi identitas, lihat [Menyediakan akses ke pengguna terautentikasi eksternal \(federasi identitas\)](#) dalam Panduan Pengguna IAM.

- Untuk mempelajari perbedaan antara menggunakan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat [Akse sumber daya lintas akun di IAM di Panduan Pengguna IAM](#).

Kontainer pod menerima kesalahan berikut: **An error occurred (SignatureDoesNotMatch) when calling the GetCallerIdentity operation: Credential should be scoped to a valid region**

Container Anda menerima kesalahan ini jika aplikasi Anda secara eksplisit membuat permintaan ke AWS STS global endpoint (<https://sts.amazonaws>) dan akun layanan Kubernetes Anda dikonfigurasi untuk menggunakan endpoint regional. Anda dapat menyelesaikan masalah dengan salah satu opsi berikut:

- Perbarui kode aplikasi Anda untuk menghapus panggilan eksplisit ke titik akhir global AWS STS.
- Perbarui kode aplikasi Anda untuk membuat panggilan eksplisit ke titik akhir regional seperti. <https://sts.us-west-2.amazonaws.com> Aplikasi Anda harus memiliki redundansi bawaan untuk memilih AWS Wilayah yang berbeda jika terjadi kegagalan layanan di AWS Wilayah. Untuk informasi selengkapnya, lihat [Mengelola AWS STS di AWS Wilayah](#) di Panduan Pengguna IAM.
- Konfigurasi akun layanan Anda untuk menggunakan titik akhir global. Cluster menggunakan endpoint regional secara default. Lihat informasi yang lebih lengkap di [the section called “Titik akhir STS”](#).

IAM role klaster Amazon EKS

Peran IAM cluster Amazon EKS diperlukan untuk setiap cluster. Cluster Kubernetes yang dikelola oleh Amazon EKS menggunakan peran ini untuk mengelola node dan [Cloud Provider Iama](#) menggunakan peran ini untuk membuat penyeimbang beban dengan Elastic Load Balancing untuk layanan.

Sebelum Anda dapat membuat klaster Amazon EKS, Anda harus membuat peran IAM dengan salah satu kebijakan IAM berikut:

- [EKSClusterKebijakan Amazon](#)
- Kebijakan IAM khusus. Izin minimal yang mengikuti memungkinkan klaster Kubernetes untuk mengelola node, tetapi tidak mengizinkan [Penyedia Cloud Iama](#) untuk membuat penyeimbang

beban dengan Elastic Load Balancing. Kebijakan IAM kustom Anda harus memiliki setidaknya izin berikut:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:CreateTags"
      ],
      "Resource": "arn:aws:ec2:*:*:instance/*",
      "Condition": {
        "ForAnyValue:StringLike": {
          "aws:TagKeys": "kubernetes.io/cluster/*"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeVpcs",
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeInstanceTopology",
        "kms:DescribeKey"
      ],
      "Resource": "*"
    }
  ]
}
```

Note

Sebelum 3 Oktober 2023, [EKSClusterKebijakan Amazon](#) diperlukan pada peran IAM untuk setiap cluster.

Sebelum 16 April 2020, [EKSServiceKebijakan Amazon](#) dan [EKSCluster Kebijakan Amazon](#) diperlukan dan nama yang disarankan untuk peran tersebut adalah `eksServiceRole`.

Dengan peran `AWSServiceRoleForAmazonEKS` terkait layanan, [EKSServicekebijakan](#)

[Kebijakan Amazon](#) tidak lagi diperlukan untuk kluster yang dibuat pada atau setelah 16 April 2020.

Periksa apakah peran kluster sudah ada

Anda dapat menggunakan prosedur berikut untuk memeriksa dan melihat apakah akun Anda sudah memiliki peran kluster Amazon EKS.

1. Buka konsol IAM di <https://console.aws.amazon.com/iam/>.
2. Di panel navigasi sebelah kiri, pilih Peran.
3. Cari daftar peran untuk eksClusterRole. Jika peran yang menyertakan eksClusterRole tidak ada, maka lihat [the section called “Membuat peran kluster Amazon EKS”](#) untuk membuat peran tersebut. Jika peran yang mencakup eksClusterRole ada, kemudian pilih peran untuk melihat kebijakan terlampir.
4. Pilih Izin.
5. Pastikan kebijakan terkelola EKSClusterKebijakan Amazon dilampirkan pada peran tersebut. Jika kebijakan terlampir, tandanya peran kluster Amazon EKS Anda dikonfigurasi dengan benar.
6. Pilih Trust relationship, lalu pilih Edit trust policy.
7. Verifikasi bahwa hubungan kepercayaan berisi kebijakan berikut. Jika hubungan kepercayaan sesuai dengan kebijakan berikut, pilih Cancel (Batalkan). Jika hubungan kepercayaan tidak cocok, salin kebijakan ke jendela Edit kebijakan kepercayaan dan pilih Perbarui kebijakan.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```


Membuat peran klaster Amazon EKS

Anda dapat menggunakan AWS Management Console atau AWS CLI untuk membuat peran cluster.

AWS Management Console

- Buka konsol IAM di <https://console.aws.amazon.com/iam/>.
- Pilih Peran, kemudian Buat peran.
- Di bawah Jenis entitas tepercaya, pilih AWS layanan.
- Dari daftar dropdown Use case for other AWS services, pilih EKS.
- Pilih EKS - Cluster untuk kasus penggunaan Anda, lalu pilih Berikutnya.
- Pada tab Tambahkan izin, pilih Berikutnya.
- Untuk nama Peran, masukkan nama unik untuk peran Anda, seperti `eksClusterRole`.
- Untuk Deskripsi, masukkan teks deskriptif seperti `Amazon EKS - Cluster role`.
- Pilih Buat peran.

AWS CLI

- Salin isi berikut ke file bernama *cluster-trust-policy.json*.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

- Buat peran. Anda dapat mengganti *eksClusterRole* dengan nama apa pun yang Anda pilih.

```
aws iam create-role \
  --role-name eksClusterRole \
  --assume-role-policy-document file://"cluster-trust-policy.json"
```

- Lampirkan kebijakan IAM yang diperlukan ke peran tersebut.

```
aws iam attach-role-policy \  
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy \  
  --role-name eksClusterRole
```

IAM role simpul Amazon EKS

kubeletDaemon simpul Amazon EKS melakukan panggilan ke AWS APIs atas nama Anda. Simpul menerima izin untuk panggilan API ini melalui profil instans IAM dan kebijakan terkait. Sebelum Anda dapat memulai node dan mendaftarkannya ke sebuah klaster, Anda harus membuat IAM role untuk node tersebut agar digunakan saat node diluncurkan. Persyaratan ini berlaku untuk node yang diluncurkan dengan AMI Amazon EKS yang dioptimalkan yang disediakan oleh Amazon, atau dengan node lain AMIs yang ingin Anda gunakan. Selain itu, persyaratan ini berlaku untuk grup node terkelola dan node yang dikelola sendiri.

Note

Anda tidak dapat menggunakan peran yang sama yang digunakan untuk membuat cluster apa pun.

Sebelum membuat node, Anda harus membuat peran IAM dengan izin berikut:

- Izin kubelet untuk mendeskripsikan EC2 sumber daya Amazon di VPC, seperti yang disediakan oleh kebijakan [Amazon EKSWorker NodePolicy](#). Kebijakan ini juga memberikan izin untuk Agen Identitas Pod Amazon EKS.
- [Izin kubelet untuk menggunakan gambar kontainer dari Amazon Elastic Container Registry \(Amazon ECR\) Registry ECR](#), seperti yang disediakan oleh kebijakan [Amazon. EC2 ContainerRegistryPullOnly](#) Izin untuk menggunakan image kontainer dari Amazon Elastic Container Registry (Amazon ECR) diperlukan karena add-on bawaan untuk pod yang menjalankan jaringan yang menggunakan image kontainer dari Amazon ECR.
- (Opsional) Izin untuk Amazon EKS Pod Identity Agent untuk menggunakan eks-auth:AssumeRoleForPodIdentity action tersebut guna mengambil kredensial Pod. Jika Anda tidak menggunakan [Amazon EKSWorker NodePolicy](#), maka Anda harus memberikan izin ini selain EC2 izin untuk menggunakan EKS Pod Identity.
- (Opsional) Jika Anda tidak menggunakan IRSA atau EKS Pod Identity untuk memberikan izin ke pod VPC CNI, maka Anda harus memberikan izin untuk VPC CNI pada peran instance. Anda dapat

menggunakan kebijakan [AmazonEKS_CNI_Policy](#)terkelola (jika Anda membuat kluster dengan IPv4 keluarga) atau [IPv6 kebijakan yang Anda buat](#) (jika Anda membuat kluster dengan IPv6 keluarga). Namun, alih-alih melampirkan kebijakan ke peran ini, sebaiknya Anda melampirkan kebijakan tersebut ke peran terpisah yang digunakan khusus untuk add-on Amazon VPC CNI. Untuk informasi selengkapnya tentang membuat peran terpisah untuk add-on Amazon VPC CNI, lihat [the section called “Konfigurasi untuk IRSA”](#)

 Note

Grup EC2 node Amazon harus memiliki peran IAM yang berbeda dari profil Fargate. Untuk informasi selengkapnya, lihat [the section called “Peran IAM eksekusi pod”](#).

Periksa apakah peran simpul sudah ada

Anda dapat menggunakan prosedur berikut untuk memeriksa dan melihat apakah akun Anda sudah memiliki peran simpul Amazon EKS.

1. Buka konsol IAM di <https://console.aws.amazon.com/iam/>.
2. Di panel navigasi sebelah kiri, pilih Peran.
3. Cari daftar peran untuk `eksNodeRole`, `AmazonEKSNodeRole`, atau `NodeInstanceRole`. Jika peran dengan salah satu nama itu tidak ada, maka lihat [the section called “Membuat IAM role simpul Amazon EKS”](#) untuk membuat peran tersebut. Jika peran yang berisi `eksNodeRole`, `AmazonEKSNodeRole`, atau `NodeInstanceRole` memang ada, maka pilih peran untuk melihat kebijakan terlampir.
4. Pilih Izin.
5. Pastikan kebijakan `EC2 ContainerRegistryPullOnly` terkelola Amazon EKSWorker NodePolicy dan Amazon dilampirkan ke peran atau kebijakan khusus dilampirkan dengan izin minimal.

 Note

Jika kebijakan `AmazonEKS_CNI_Policy` terlampir pada peran, sebaiknya hapus dan lampirkan kebijakan tersebut ke IAM role yang dipetakan ke akun layanan `aws-node-kubernetes` sebagai gantinya. Untuk informasi selengkapnya, lihat [the section called “Konfigurasi untuk IRSA”](#).

6. Pilih Trust relationship, lalu pilih Edit trust policy.

7. Verifikasi bahwa hubungan kepercayaan berisi kebijakan berikut. Jika hubungan kepercayaan sesuai dengan kebijakan berikut, pilih Cancel (Batalkan). Jika hubungan kepercayaan tidak cocok, salin kebijakan ke jendela Edit kebijakan kepercayaan dan pilih Perbarui kebijakan.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
      ],
      "Principal": {
        "Service": [
          "ec2.amazonaws.com"
        ]
      }
    }
  ]
}
```

Membuat IAM role simpel Amazon EKS

Anda dapat membuat peran IAM node dengan AWS Management Console atau AWS CLI.

AWS Management Console

- Buka konsol IAM di <https://console.aws.amazon.com/iam/>.
- Di panel navigasi sebelah kiri, pilih Peran.
- Pada halaman Peran, pilih Buat peran.
- Pada halaman Pilih entitas tepercaya, lakukan hal berikut:
 - Di bagian Jenis entitas tepercaya, pilih AWS layanan.
 - Di bawah Kasus penggunaan, pilih EC2.
 - Pilih Berikutnya.
- Pada halaman Tambahkan izin, lampirkan kebijakan khusus atau lakukan hal berikut:
 - Di dalam kotak Filter kebijakan, masukkan AmazonEKSEKWorkerNodePolicy.
 - Pilih kotak centang di sebelah kiri Amazon EKSEKWorker NodePolicy di hasil pencarian.
 - Pilih Hapus filter.

- iv. Di dalam kotak Filter kebijakan, masukkan AmazonEC2ContainerRegistryPullOnly.
- v. Pilih kotak centang di sebelah kiri Amazon EC2 ContainerRegistryPullOnly di hasil pencarian.

[Kebijakan terkelola Amazoneks_CNI_Policy](#), atau kebijakan yang Anda buat juga harus dilampirkan pada peran ini atau IPv6 peran lain yang dipetakan ke akun layanan

[Kubernetes](#). aws-node Sebaiknya tetapkan kebijakan ke peran yang terkait dengan akun layanan Kubernetes, alih-alih menugaskannya ke peran ini. Untuk informasi selengkapnya, lihat [the section called “Konfigurasi untuk IRSA”](#).

- vi. Pilih Berikutnya.
- f. Pada halaman Nama, tinjau, dan buat, lakukan hal berikut:
 - i. Untuk nama Peran, masukkan nama unik untuk peran Anda, seperti AmazonEKSNodeRole.
 - ii. Untuk Deskripsi, ganti teks saat ini dengan teks deskriptif seperti Amazon EKS - Node role.
 - iii. Di bawah Tambahkan tag (Opsional), tambahkan metadata ke peran dengan melampirkan tag sebagai pasangan nilai kunci. Untuk informasi selengkapnya tentang penggunaan tanda di IAM, lihat [Menandai sumber daya IAM](#) di Panduan Pengguna IAM.
- iv. Pilih Buat peran.

AWS CLI

- a. Jalankan perintah berikut untuk membuat node-role-trust-relationship.json file.

```
cat >node-role-trust-relationship.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
      ],
      "Principal": {
        "Service": [
          "ec2.amazonaws.com"
        ]
      }
    }
  ]
}
```

EOF

b. Buat peran IAM.

```
aws iam create-role \  
  --role-name AmazonEKSNodeRole \  
  --assume-role-policy-document file://"node-role-trust-relationship.json"
```

c. Lampirkan dua kebijakan yang dikelola IAM yang diperlukan ke peran IAM.

```
aws iam attach-role-policy \  
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSWorkerNodePolicy \  
  --role-name AmazonEKSNodeRole  
aws iam attach-role-policy \  
  --policy-arn arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryPullOnly \  
  --role-name AmazonEKSNodeRole
```

d. Lampirkan salah satu kebijakan IAM berikut ke peran IAM tergantung pada keluarga IP mana Anda membuat kluster. Kebijakan harus dilampirkan pada peran ini atau ke peran yang terkait dengan akun aws-node layanan Kubernetes yang digunakan untuk plugin Amazon VPC CNI untuk Kubernetes. Sebaiknya tugaskan kebijakan ke dalam peran yang terkait dengan akun layanan Kubernetes. Untuk menetapkan kebijakan ke peran yang terkait dengan akun layanan Kubernetes, lihat. [the section called “Konfigurasi untuk IRSA”](#)

- IPv4

```
aws iam attach-role-policy \  
  --policy-arn arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy \  
  --role-name AmazonEKSNodeRole
```

- IPv6

A. Salin teks berikut dan simpan ke file bernama `vpc-cni-ipv6-policy.json`.

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "ec2:AssignIpv6Addresses",  
        "ec2:DescribeInstances",  
        "ec2:DescribeTags",
```

```

        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeInstanceTypes"
    ],
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "ec2:CreateTags"
    ],
    "Resource": [
      "arn:aws:ec2:*:*:network-interface/*"
    ]
  }
]
}

```

B. Buat kebijakan IAM.

```
aws iam create-policy --policy-name AmazonEKS_CNI_IPv6_Policy --policy-document file://vpc-cni-ipv6-policy.json
```

C. Lampirkan kebijakan IAM ke peran IAM. Ganti **111122223333** dengan ID akun Anda.

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::111122223333:policy/AmazonEKS_CNI_IPv6_Policy \
  --role-name AmazonEKSNodeRole
```

Peran IAM kluster Mode Otomatis Amazon EKS

Peran IAM cluster Amazon EKS diperlukan untuk setiap cluster. Cluster Kubernetes yang dikelola oleh Amazon EKS menggunakan peran ini untuk mengotomatiskan tugas rutin untuk penyimpanan, jaringan, dan komputasi penskalaan otomatis.

Sebelum Anda dapat membuat kluster Amazon EKS, Anda harus membuat peran IAM dengan kebijakan yang diperlukan untuk Mode Otomatis EKS. Anda dapat melampirkan kebijakan terkelola AWS IAM yang disarankan, atau membuat kebijakan khusus dengan izin yang setara.

- [EKSCComputeKebijakan Amazon](#)
- [Amazon EKSBBlock StoragePolicy](#)

- [Amazon EKSLoad BalancingPolicy](#)
- [EKSNetworkingKebijakan Amazon](#)
- [EKSClusterKebijakan Amazon](#)

Periksa apakah peran kluster sudah ada

Anda dapat menggunakan prosedur berikut untuk memeriksa dan melihat apakah akun Anda sudah memiliki peran kluster Amazon EKS.

1. Buka konsol IAM di <https://console.aws.amazon.com/iam/>.
2. Di panel navigasi sebelah kiri, pilih Peran.
3. Cari daftar peran untuk AmazonEKSAutoClusterRole. Jika peran yang menyertakan AmazonEKSAutoClusterRole tidak ada, lihat instruksi di bagian berikutnya untuk membuat peran. Jika peran yang mencakup AmazonEKSAutoClusterRole ada, kemudian pilih peran untuk melihat kebijakan terlampir.
4. Pilih Izin.
5. Pastikan kebijakan terkelola EKSClusterKebijakan Amazon dilampirkan pada peran tersebut. Jika kebijakan terlampir, tandanya peran kluster Amazon EKS Anda dikonfigurasi dengan benar.
6. Pilih Trust relationship, lalu pilih Edit trust policy.
7. Verifikasi bahwa hubungan kepercayaan berisi kebijakan berikut. Jika hubungan kepercayaan sesuai dengan kebijakan berikut, pilih Cancel (Batalkan). Jika hubungan kepercayaan tidak cocok, salin kebijakan ke jendela Edit kebijakan kepercayaan dan pilih Perbarui kebijakan.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```



```
}
```

Note

AWS tidak memerlukan nama `AmazonEKSAutoClusterRole` untuk peran ini.

Membuat peran klaster Amazon EKS

Anda dapat menggunakan AWS Management Console atau AWS CLI untuk membuat peran cluster.

AWS Management Console

1. Buka konsol IAM di <https://console.aws.amazon.com/iam/>.
2. Pilih Peran, kemudian Buat peran.
3. Di bawah Jenis entitas tepercaya, pilih AWS layanan.
4. Dari daftar dropdown Use case for other AWS services, pilih EKS.
5. Pilih EKS - Cluster untuk kasus penggunaan Anda, lalu pilih Berikutnya.
6. Pada tab Tambahkan izin, pilih kebijakan, lalu pilih Berikutnya.
 - [EKSCComputeKebijakan Amazon](#)
 - [Amazon EKSBLOCK StoragePolicy](#)
 - [Amazon EKSLoad BalancingPolicy](#)
 - [EKSNetworkingKebijakan Amazon](#)
 - [EKSClusterKebijakan Amazon](#)
7. Untuk nama Peran, masukkan nama unik untuk peran Anda, seperti `AmazonEKSAutoClusterRole`.
8. Untuk Deskripsi, masukkan teks deskriptif seperti `Amazon EKS - Cluster role`.
9. Pilih Buat peran.

AWS CLI

1. Salin isi berikut ke file bernama *cluster-trust-policy.json*.

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Principal": {
      "Service": "eks.amazonaws.com"
    },
    "Action": [
      "sts:AssumeRole",
      "sts:TagSession"
    ]
  }
]
```

2. Buat peran. Anda dapat mengganti *AmazonEKSAutoClusterRole* dengan nama apa pun yang Anda pilih.

```
aws iam create-role \
  --role-name AmazonEKSAutoClusterRole \
  --assume-role-policy-document file://"cluster-trust-policy.json"
```

3. Lampirkan kebijakan IAM yang diperlukan ke peran:

EKSClusterKebijakan Amazon:

```
aws iam attach-role-policy \
  --role-name AmazonEKSAutoClusterRole \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSClusterPolicy
```

EKSComputeKebijakan Amazon:

```
aws iam attach-role-policy \
  --role-name AmazonEKSAutoClusterRole \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSClusterPolicy
```

Amazon EKSBLOCK StoragePolicy:

```
aws iam attach-role-policy \
  --role-name AmazonEKSAutoClusterRole \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSBLOCKStoragePolicy
```

Amazon EKSLoad BalancingPolicy:

```
aws iam attach-role-policy \  
  --role-name AmazonEKSAutoClusterRole \  
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSLoadBalancingPolicy
```

EKSNetworkingKebijakan Amazon:

```
aws iam attach-role-policy \  
  --role-name AmazonEKSAutoClusterRole \  
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSNetworkingPolicy
```

Peran IAM simpul Mode Otomatis Amazon EKS

Note

Anda tidak dapat menggunakan peran yang sama yang digunakan untuk membuat cluster apa pun.

Sebelum membuat node, Anda harus membuat peran IAM dengan kebijakan berikut, atau izin yang setara:

- [Amazon EKSWorker NodeMinimalPolicy](#)
- [Amazon EC2 ContainerRegistryPullOnly](#)

Periksa apakah peran simpul sudah ada

Anda dapat menggunakan prosedur berikut untuk memeriksa dan melihat apakah akun Anda sudah memiliki peran simpul Amazon EKS.

1. Buka konsol IAM di <https://console.aws.amazon.com/iam/>.
2. Di panel navigasi sebelah kiri, pilih Peran.
3. Cari daftar peran untuk AmazonEKSAutoNodeRole. Jika peran dengan salah satu nama itu tidak ada, lihat instruksi di bagian berikutnya untuk membuat peran. Jika peran yang berisi AmazonEKSAutoNodeRole memang ada, pilih peran untuk melihat kebijakan terlampir.
4. Pilih Izin.

5. Pastikan bahwa kebijakan yang diperlukan di atas terlampir, atau kebijakan khusus yang setara.
6. Pilih Trust relationship, lalu pilih Edit trust policy.
7. Verifikasi bahwa hubungan kepercayaan berisi kebijakan berikut. Jika hubungan kepercayaan sesuai dengan kebijakan berikut, pilih Cancel (Batalkan). Jika hubungan kepercayaan tidak cocok, salin kebijakan ke jendela Edit kebijakan kepercayaan dan pilih Perbarui kebijakan.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Membuat IAM role simpel Amazon EKS

Anda dapat membuat peran IAM node dengan AWS Management Console atau AWS CLI.

AWS Management Console

1. Buka konsol IAM di <https://console.aws.amazon.com/iam/>.
2. Di panel navigasi sebelah kiri, pilih Peran.
3. Pada halaman Peran, pilih Buat peran.
4. Pada halaman Pilih entitas tepercaya, lakukan hal berikut:
 - a. Di bagian Jenis entitas tepercaya, pilih AWS layanan.
 - b. Di bawah Kasus penggunaan, pilih EC2.
 - c. Pilih Berikutnya.
5. Pada halaman Tambahkan izin, lampirkan kebijakan berikut:
 - [Amazon EKSWorker NodeMinimalPolicy](#)
 - [Amazon EC2 ContainerRegistryPullOnly](#)
6. Pada halaman Nama, tinjau, dan buat, lakukan hal berikut:

- a. Untuk nama Peran, masukkan nama unik untuk peran Anda, seperti `AmazonEKSAutoNodeRole`.
- b. Untuk Deskripsi, ganti teks saat ini dengan teks deskriptif seperti `Amazon EKS - Node role`.
- c. Di bawah Tambahkan tag (Opsional), tambahkan metadata ke peran dengan melampirkan tag sebagai pasangan nilai kunci. Untuk informasi selengkapnya tentang penggunaan tanda di IAM, lihat [Menandai sumber daya IAM](#) di Panduan Pengguna IAM.
- d. Pilih Buat peran.

AWS CLI

Buat Peran IAM Node

Gunakan `node-trust-policyfile.json` dari langkah sebelumnya untuk menentukan entitas mana yang dapat mengambil peran. Jalankan perintah berikut untuk membuat Peran IAM Node:

```
aws iam create-role \  
  --role-name AmazonEKSAutoNodeRole \  
  --assume-role-policy-document file://node-trust-policy.json
```

Perhatikan Peran ARN

Setelah membuat peran, ambil dan simpan ARN dari Peran IAM Node. Anda akan membutuhkan ARN ini pada langkah selanjutnya. Gunakan perintah berikut untuk mendapatkan ARN:

```
aws iam get-role --role-name AmazonEKSAutoNodeRole --query "Role.Arn" --output text
```

Lampirkan Kebijakan yang Diperlukan

Lampirkan kebijakan AWS terkelola berikut ke Peran IAM Node untuk memberikan izin yang diperlukan:

Untuk melampirkan `AmazonEKSWorkerNodeMinimalPolicy`:

```
aws iam attach-role-policy \  
  --role-name AmazonEKSAutoNodeRole \  
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodeMinimalPolicy
```

Untuk melampirkan `AmazonEC2ContainerRegistryPullOnly`:

```
aws iam attach-role-policy \  
  --role-name AmazonEKSAutoNodeRole \  
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryPullOnly
```

Pantau kinerja klaster Anda dan lihat log

Anda dapat mengamati data Anda di Amazon EKS menggunakan banyak alat pemantauan atau pencatatan yang tersedia. Data log Amazon EKS Anda dapat dialirkan ke AWS layanan atau ke alat mitra untuk analisis data. Ada banyak layanan yang tersedia di AWS Management Console yang menyediakan data untuk memecahkan masalah Amazon EKS Anda. Anda juga dapat menggunakan solusi sumber terbuka yang AWS didukung untuk memantau infrastruktur [Amazon EKS](#).

Setelah memilih Cluster di panel navigasi kiri konsol Amazon EKS, Anda dapat melihat kesehatan dan detail klaster dengan memilih nama cluster dan memilih tab Observability. Untuk melihat detail tentang resource Kubernetes yang ada yang di-deploy ke klaster Anda, lihat. [the section called “Akses sumber daya klaster”](#)

Pemantauan adalah bagian penting dalam menjaga keandalan, ketersediaan, dan kinerja Amazon EKS dan AWS solusi Anda. Kami menyarankan Anda mengumpulkan data pemantauan dari semua bagian AWS solusi Anda. Dengan begitu, Anda dapat lebih mudah men-debug kegagalan multi-titik jika terjadi. Sebelum Anda mulai memantau Amazon EKS, pastikan bahwa rencana pemantauan Anda menjawab pertanyaan-pertanyaan berikut.

- Apa tujuan Anda? Apakah Anda memerlukan pemberitahuan waktu nyata jika skala cluster Anda dramatis?
- Sumber daya apa yang perlu diamati?
- Seberapa sering Anda perlu mengamati sumber daya ini? Apakah perusahaan Anda ingin merespons risiko dengan cepat?
- Alat apa yang ingin Anda gunakan? Jika Anda sudah menjalankan AWS Fargate sebagai bagian dari peluncuran Anda, maka Anda dapat menggunakan router [log](#) bawaan.
- Siapa yang ingin Anda lakukan untuk melakukan tugas pemantauan?
- Kepada siapa Anda ingin pemberitahuan dikirim ketika terjadi kesalahan?

Pemantauan dan pencatatan di Amazon EKS

Amazon EKS menyediakan alat bawaan untuk pemantauan dan pencatatan. Untuk versi yang didukung, dasbor observabilitas memberikan visibilitas ke kinerja klaster Anda. Ini membantu Anda mendeteksi, memecahkan masalah, dan memperbaiki masalah dengan cepat. Selain fitur pemantauan, ini mencakup daftar berdasarkan log audit bidang kontrol. Bidang kontrol Kubernetes memperlihatkan sejumlah metrik yang juga dapat dikikis di luar konsol.

Pencatatan pesawat kontrol merekam semua panggilan API ke kluster Anda, mengaudit informasi yang menangkap pengguna yang melakukan tindakan apa pada kluster Anda, dan informasi berbasis peran. Untuk informasi selengkapnya, lihat [Pencatatan dan pemantauan di Amazon EKS](#) di Panduan AWS Preskriptif.

Pencatatan pesawat kontrol Amazon EKS menyediakan log audit dan diagnostik langsung dari bidang kontrol Amazon EKS ke CloudWatch Log di akun Anda. Log ini memudahkan Anda untuk mengamankan dan menjalankan kluster Anda. Anda dapat memilih jenis log yang tepat yang Anda butuhkan, dan log dikirim sebagai aliran log ke grup untuk setiap kluster Amazon EKS. CloudWatch Untuk informasi selengkapnya, lihat [the section called “Kontrol log pesawat”](#).

Note

Saat Anda memeriksa log autentikator Amazon EKS di Amazon CloudWatch, entri ditampilkan yang berisi teks yang mirip dengan teks contoh berikut.

```
level=info msg="mapping IAM role" groups="[]" role="arn:aws:iam::111122223333:role/XXXXXXXXXXXXXXXXXXXX-NodeManagerRole-XXXXXXX"
username="eks:node-manager"
```

Entri yang berisi teks ini diharapkan. `username` adalah peran layanan internal Amazon EKS yang melakukan operasi spesifik untuk grup simpul terkelola dan Fargate.

Untuk logging tingkat rendah dan dapat disesuaikan, maka logging [Kubernetes](#) tersedia.

Amazon EKS terintegrasi dengan AWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan di Amazon EKS. CloudTrail menangkap semua panggilan API untuk Amazon EKS sebagai acara. Panggilan yang tertangkap meliputi panggilan dari konsol Amazon EKS dan panggilan kode ke operasi API Amazon EKS. Untuk informasi selengkapnya, lihat [the section called “AWS CloudTrail”](#).

Server Kubernetes API menampilkan sejumlah metrik yang berguna untuk pemantauan dan analisis. Untuk informasi selengkapnya, lihat [the section called “Metrik Prometheus”](#).

Untuk mengonfigurasi Bit Lancar untuk CloudWatch log Amazon kustom, lihat [Menyiapkan Bit Lancar](#) di CloudWatch Panduan Pengguna Amazon.

Alat pemantauan dan pencatatan Amazon EKS

Amazon Web Services menyediakan berbagai alat yang dapat Anda gunakan untuk memantau Amazon EKS. Anda dapat mengonfigurasi beberapa alat untuk mengatur pemantauan otomatis, tetapi beberapa memerlukan panggilan manual. Kami menyarankan Anda mengotomatiskan tugas pemantauan sebanyak yang diizinkan oleh lingkungan dan toolset yang ada.

Tabel berikut menjelaskan berbagai opsi alat pemantauan.

Area	Alat	Deskripsi	Pengaturan
Bidang kontrol	Dasbor observabilitas	Untuk versi yang didukung, dasbor observabilitas memberikan visibilitas ke kinerja kluster Anda. Ini membantu Anda mendeteksi, memecahkan masalah, dan memperbaiki masalah dengan cepat.	Prosedur pengaturan
Aplikasi/bidang kontrol	Prometheus	Prometheus dapat digunakan untuk memantau metrik dan peringatan untuk aplikasi dan bidang kontrol.	Prosedur pengaturan
Aplikasi	CloudWatch Wawasan Kontainer	CloudWatch Container Insights mengumpulkan, mengumpulkan, dan merangkum metrik dan log dari aplikasi dan layanan mikro	Prosedur pengaturan

Area	Alat	Deskripsi	Pengaturan
		dalam kontainer Anda.	
Aplikasi	AWS Distro untuk OpenTelemetry (ADOT)	ADOT dapat mengumpulkan dan mengirim an metrik yang berkorelasi, melacak data, dan metadata ke AWS layanan pemantauan atau mitra. Itu dapat diatur melalui CloudWatch Wawasan Kontainer.	Prosedur pengaturan
Aplikasi	DevOpsGuru Amazon	Amazon DevOps Guru mendeteksi kinerja dan ketersediaan operasional tingkat simpul.	Prosedur pengaturan
Aplikasi	AWS X-Ray	AWS X-Ray menerima data jejak tentang aplikasi Anda. Data jejak ini mencakup permintaan masuk dan keluar serta metadata tentang permintaan. Untuk Amazon EKS, implementasinya memerlukan OpenTelemetry add-on.	Prosedur pengaturan

Area	Alat	Deskripsi	Pengaturan
Aplikasi	Amazon CloudWatch	CloudWatch menyediakan beberapa metrik Amazon EKS dasar secara gratis pada versi yang didukung. Anda dapat memperluas fungsionalitas ini dengan Operator CloudWatch Observabilitas untuk menangani pengumpulan metrik, log, dan data penelusuran.	Prosedur pengaturan

Tabel berikut menjelaskan berbagai opsi alat logging.

Area	Alat	Deskripsi	Pengaturan
Bidang kontrol	Dasbor observabilitas	Untuk versi yang didukung, dasbor observabilitas menampilkan daftar berdasarkan log audit bidang kontrol. Ini juga mencakup tautan untuk mengontrol log pesawat di Amazon CloudWatch.	Prosedur pengaturan

Area	Alat	Deskripsi	Pengaturan
Aplikasi	CloudWatch Wawasan Kontainer Amazon	Amazon CloudWatch Container Insights mengumpulkan, menggabungkan, dan merangkum metrik dan log dari aplikasi container dan layanan mikro Anda.	Prosedur pengaturan
Bidang kontrol	CloudWatch Log Amazon	Anda dapat mengirim log audit dan diagnostik langsung dari bidang kontrol Amazon EKS ke CloudWatch Log di akun Anda.	Prosedur pengaturan
Bidang kontrol	AWS CloudTrail	Ini mencatat panggilan API oleh pengguna, peran, atau layanan.	Prosedur pengaturan
Beberapa area untuk instance AWS Fargate	AWS Router log Fargate	Untuk instance AWS Fargate, router log mengalirkan log ke AWS layanan atau alat mitra. Ini digunakan AWS untuk Fluent Bit . Log dapat dialirkan ke AWS layanan lain atau alat mitra.	Prosedur pengaturan

Pantau klaster Anda dengan dasbor observabilitas

Konsol Amazon EKS menyertakan dasbor observabilitas yang memberikan visibilitas ke kinerja klaster Anda. Informasi yang diberikannya membantu Anda mendeteksi, memecahkan masalah, dan memperbaiki masalah dengan cepat. Anda dapat membuka bagian dasbor observabilitas yang berlaku dengan memilih item di ringkasan Kesehatan dan kinerja. Ringkasan ini disertakan di beberapa tempat, termasuk tab Observability.

Dasbor observabilitas dibagi menjadi beberapa tab.

Ringkasan

Ringkasan Kesehatan dan kinerja mencantumkan jumlah item dalam berbagai kategori. Setiap nomor bertindak sebagai hyperlink ke lokasi di dasbor observabilitas dengan daftar untuk kategori itu.

Kesehatan cluster

Kesehatan cluster menyediakan pemberitahuan penting yang harus diperhatikan, beberapa di antaranya mungkin perlu Anda ambil tindakan sesegera mungkin. Dengan daftar ini, Anda dapat melihat deskripsi dan sumber daya yang terpengaruh. Kesehatan cluster mencakup dua tabel: Masalah Kesehatan dan wawasan Konfigurasi. Untuk menyegarkan status masalah Kesehatan, pilih tombol refresh (). Wawasan konfigurasi diperbarui secara otomatis setiap 24 jam sekali dan tidak dapat disegarkan secara manual.

Untuk informasi lebih lanjut tentang masalah Kesehatan, lihat [the section called “Kesehatan cluster FAQs dan kode kesalahan dengan jalur resolusi”](#). Untuk informasi selengkapnya tentang wawasan Konfigurasi, lihat [the section called “Wawasan cluster”](#).

Kontrol pemantauan pesawat

Tab pemantauan bidang kontrol dibagi menjadi tiga bagian, yang masing-masing membantu Anda memantau dan memecahkan masalah bidang kontrol cluster Anda.

Metrik

Untuk cluster yang versi 1.28 Kubernetes ke atas, bagian Metrik menunjukkan grafik dari beberapa metrik yang dikumpulkan untuk berbagai komponen bidang kontrol.

Anda dapat mengatur periode waktu yang digunakan oleh sumbu X dari setiap grafik dengan membuat pilihan di bagian atas bagian. Anda dapat menyegarkan data dengan tombol refresh (). Untuk setiap grafik terpisah, tombol elips vertikal () membuka menu dengan opsi dari CloudWatch

Metrik ini dan lainnya secara otomatis tersedia sebagai metrik pemantauan dasar di CloudWatch bawah namespace. AWS/EKS Untuk informasi selengkapnya, lihat [Pemantauan dasar dan pemantauan terperinci](#) di Panduan CloudWatch Pengguna Amazon. Untuk mendapatkan metrik, visualisasi, dan wawasan yang lebih mendetail, lihat Wawasan [Penampung di Panduan Pengguna Amazon](#). CloudWatch Atau jika Anda lebih suka pemantauan berbasis Prometheus, lihat. [the section called “Metrik Prometheus”](#)

Tabel berikut menjelaskan metrik yang tersedia.

Metrik	Deskripsi
APIServer Permintaan	Permintaan per menit dibuat ke server API.
APIServer Total Permintaan 4XX	Hitungan permintaan server API per menit yang memiliki kode respons HTTP 4XX (kesalahan sisi klien).
APIServer Total Permintaan 5XX	Hitungan permintaan server API per menit yang memiliki kode respons HTTP 5XX (kesalahan sisi server).
APIServer Total Permintaan 429	Hitungan permintaan server API per menit yang memiliki kode respons HTTP 429 (terlalu banyak permintaan).
Ukuran penyimpanan	Ukuran database penyimpanan (etcd).
Upaya penjadwal	Jumlah upaya untuk menjadwalkan pod berdasarkan hasil “unschedulable” “error”, dan “scheduled”.
Pod yang tertunda	Jumlah pod yang tertunda menurut tipe antrian “active”, “backoff”, “unschedulable”, dan “gated”.

Metrik	Deskripsi
Latensi permintaan server API	Latensi untuk permintaan server API.
Permintaan dalam pesawat server API saat ini	Permintaan dalam penerbangan saat ini untuk server API.
Permintaan Webhook	Permintaan webhook per menit.
Penolakan permintaan Webhook	Hitungan permintaan webhook yang ditolak.
Latensi permintaan Webhook P99	Latensi persentil ke-99 dari permintaan webhook pihak ketiga eksternal.

CloudWatch Wawasan Log

Bagian Wawasan CloudWatch Log menampilkan berbagai daftar berdasarkan log audit bidang kontrol. Log bidang kontrol Amazon EKS perlu dihidupkan untuk menggunakan fitur ini, yang dapat Anda lakukan dari bagian Lihat log bidang kontrol di CloudWatch bagian.

Ketika cukup waktu telah berlalu untuk mengumpulkan data, Anda dapat Menjalankan semua kueri atau memilih Jalankan kueri untuk satu daftar pada satu waktu. Biaya tambahan akan dikenakan dari CloudWatch setiap kali Anda menjalankan kueri. Pilih periode waktu hasil yang ingin Anda lihat di bagian atas bagian. Jika Anda ingin kontrol yang lebih canggih untuk kueri apa pun, Anda dapat memilih Lihat di CloudWatch. Ini akan memungkinkan Anda memperbarui kueri CloudWatch agar sesuai dengan kebutuhan Anda.

Untuk informasi selengkapnya, lihat [Menganalisis data CloudWatch log dengan Wawasan Log](#) di Panduan Pengguna CloudWatch Log Amazon.

Lihat log bidang kontrol di CloudWatch

Pilih Kelola logging untuk memperbarui jenis log yang tersedia. Dibutuhkan beberapa menit agar log muncul di CloudWatch Log setelah Anda mengaktifkan logging. Ketika cukup waktu telah berlalu, pilih salah satu tautan Lihat di bagian ini untuk menavigasi ke log yang berlaku.

Untuk informasi selengkapnya, lihat [the section called “Kontrol log pesawat”](#).

Wawasan cluster

Tabel Upgrade insights memunculkan masalah dan merekomendasikan tindakan korektif, mempercepat proses validasi untuk upgrade ke versi Kubernetes baru. Amazon EKS secara otomatis memindai kluster terhadap daftar potensi peningkatan versi Kubernetes yang berdampak pada masalah. Tabel Upgrade insights mencantumkan pemeriksaan insight yang dilakukan oleh Amazon EKS terhadap kluster ini, beserta status terkaitnya.

Amazon EKS memelihara dan secara berkala menyegarkan daftar pemeriksaan wawasan yang akan dilakukan berdasarkan evaluasi perubahan dalam proyek Kubernetes serta perubahan layanan Amazon EKS yang terkait dengan versi baru. Konsol Amazon EKS secara otomatis menyegarkan status setiap wawasan, yang dapat dilihat di kolom waktu penyegaran terakhir.

Untuk informasi selengkapnya, lihat [the section called “Wawasan cluster”](#).

Masalah kesehatan simpul

Agen pemantauan simpul Amazon EKS secara otomatis membaca log node untuk mendeteksi masalah kesehatan. Terlepas dari pengaturan perbaikan otomatis, semua masalah kesehatan node dilaporkan sehingga Anda dapat menyelidiki sesuai kebutuhan. Jika jenis masalah terdaftar tanpa deskripsi, Anda dapat membaca deskripsi di elemen popovernya.

Saat Anda me-refresh halaman, masalah yang diselesaikan akan hilang dari daftar. Jika perbaikan otomatis diaktifkan, Anda sementara dapat melihat beberapa masalah kesehatan yang akan diselesaikan tanpa tindakan dari Anda. Masalah yang tidak didukung oleh perbaikan otomatis mungkin memerlukan tindakan manual dari Anda tergantung pada jenisnya.

Agar masalah kesehatan node dilaporkan, kluster Anda harus menggunakan Amazon EKS Auto Mode atau memiliki add-on agen pemantauan node. Lihat informasi yang lebih lengkap di [the section called “Kesehatan simpul”](#).

Pantau metrik kluster Anda dengan Prometheus

[Prometheus](#) adalah database pemantauan dan deret waktu yang menggores titik akhir. Ini memberikan kemampuan untuk query, agregat, dan menyimpan data yang dikumpulkan. Anda juga dapat menggunakannya untuk peringatan dan agregasi peringatan. Topik ini menjelaskan cara mengatur Prometheus sebagai opsi terkelola atau open source. Memantau metrik bidang kontrol Amazon EKS adalah kasus penggunaan yang umum.

Amazon Managed Service for Prometheus adalah layanan pemantauan dan peringatan Prometheus yang kompatibel dengan Prometheus yang memudahkan pemantauan aplikasi dan infrastruktur dalam skala besar. Ini adalah layanan yang dikelola sepenuhnya yang secara otomatis menskalakan konsumsi, penyimpanan, kueri, dan peringatan metrik Anda. Ini juga terintegrasi dengan layanan AWS keamanan untuk memungkinkan akses cepat dan aman ke data Anda. Anda dapat menggunakan bahasa kueri PromQL sumber terbuka untuk menanyakan metrik dan memperingatkannya. Selain itu, Anda dapat menggunakan pengelola peringatan di Amazon Managed Service untuk Prometheus untuk menyiapkan aturan peringatan untuk peringatan penting. Anda kemudian dapat mengirim peringatan penting ini sebagai pemberitahuan ke topik Amazon SNS.

Ada beberapa opsi berbeda untuk menggunakan Prometheus dengan Amazon EKS:

- Anda dapat mengaktifkan metrik Prometheus saat pertama kali membuat cluster Amazon EKS atau Anda dapat membuat scraper Prometheus sendiri untuk cluster yang ada. Kedua opsi ini dicakup oleh topik ini.
- Anda dapat menggunakan Prometheus menggunakan Helm. Untuk informasi selengkapnya, lihat [the section called “Terapkan menggunakan Helm”](#).
- Anda dapat melihat metrik mentah bidang kontrol dalam format Prometheus. Untuk informasi selengkapnya, lihat [the section called “Bidang kontrol”](#).

Langkah 1: Aktifkan metrik Prometheus

Important

Layanan Terkelola Amazon untuk sumber daya Prometheus berada di luar siklus hidup klaster dan perlu dipertahankan secara independen dari klaster. Saat Anda menghapus klaster Anda, pastikan juga untuk menghapus pencakar yang berlaku untuk menghentikan biaya yang berlaku. Untuk informasi selengkapnya, lihat [Menemukan dan menghapus pencakar](#) di Amazon Managed Service for Prometheus User Guide.

Prometheus menemukan dan mengumpulkan metrik dari cluster Anda melalui model berbasis tarik yang disebut scraping. Scraper disiapkan untuk mengumpulkan data dari infrastruktur cluster dan aplikasi kontainer Anda. Saat Anda mengaktifkan opsi untuk mengirim metrik Prometheus, Layanan Terkelola Amazon untuk Prometheus menyediakan pengikis tanpa agen yang dikelola sepenuhnya.

Jika Anda belum membuat cluster, Anda dapat mengaktifkan opsi untuk mengirim metrik ke Prometheus saat pertama kali membuat cluster. Di konsol Amazon EKS, opsi ini ada di langkah Konfigurasi observabilitas untuk membuat cluster baru. Untuk informasi selengkapnya, lihat [the section called “Membuat klaster”](#).

Jika Anda sudah memiliki cluster yang ada, Anda dapat membuat scraper Prometheus Anda sendiri. Untuk melakukan ini di konsol Amazon EKS, navigasikan ke tab Observability cluster Anda dan pilih tombol Add scraper. Jika Anda lebih suka melakukannya dengan AWS API atau AWS CLI, lihat [Membuat scraper](#) di Amazon Managed Service for Prometheus User Guide.

Opsi berikut tersedia saat membuat scraper dengan konsol Amazon EKS.

Scraper alias

(Opsional) Masukkan alias unik untuk scraper.

Destinasi

Pilih Layanan Terkelola Amazon untuk ruang kerja Prometheus. Ruang kerja adalah ruang logis yang didedikasikan untuk penyimpanan dan kueri metrik Prometheus. Dengan ruang kerja ini, Anda akan dapat melihat metrik Prometheus di seluruh akun yang memiliki akses ke sana. Opsi Buat ruang kerja baru memberi tahu Amazon EKS untuk membuat ruang kerja atas nama Anda menggunakan alias Workspace yang Anda berikan. Dengan opsi Pilih ruang kerja yang ada, Anda dapat memilih ruang kerja yang ada dari daftar tarik-turun. Untuk informasi selengkapnya tentang ruang kerja, lihat [Mengelola ruang kerja](#) di Amazon Managed Service for Prometheus User Guide.

Akses layanan

Bagian ini merangkum izin yang Anda berikan saat mengirim metrik Prometheus:

- Izinkan Layanan Terkelola Amazon untuk Prometheus mendeskripsikan kluster Amazon EKS yang tergores
- Izinkan penulisan jarak jauh ke ruang kerja Prometheus yang Dikelola Amazon

Jika AmazonManagedScraperRole sudah ada, scraper menggunakannya.

Pilih AmazonManagedScraperRole tautan untuk melihat detail Izin. Jika

AmazonManagedScraperRole belum ada, pilih tautan Lihat detail izin untuk melihat izin tertentu yang Anda berikan dengan mengirimkan metrik Prometheus.

Subnet

Ubah subnet yang akan diwarisi scraper sesuai kebutuhan. Jika Anda perlu menambahkan opsi subnet berwarna abu-abu, kembali ke langkah buat cluster Tentukan jaringan.

Konfigurasi scraper

Ubah konfigurasi scraper dalam format YAMAL sesuai kebutuhan. Untuk melakukannya, gunakan formulir atau unggah file YAMM pengganti. Untuk informasi selengkapnya, lihat [Konfigurasi pengikis](#) di Amazon Managed Service for Prometheus User Guide.

Amazon Managed Service untuk Prometheus mengacu pada scraper tanpa agen yang dibuat bersama cluster sebagai kolektor terkelola. AWS Untuk informasi selengkapnya tentang kolektor AWS terkelola, lihat [Metrik konsumsi dengan kolektor AWS terkelola di](#) Panduan Pengguna Layanan Terkelola Amazon untuk Prometheus.

Important

- Jika Anda membuat scraper Prometheus menggunakan AWS CLI atau API, Anda perlu menyesuaikan konfigurasinya untuk memberikan izin dalam cluster scraper. Untuk informasi selengkapnya, lihat [Mengonfigurasi klaster Amazon EKS Anda](#) di Panduan Pengguna Layanan Terkelola Amazon untuk Prometheus.
- Jika Anda memiliki scraper Prometheus yang dibuat sebelum 11 November 2024 yang menggunakan `aws-auth ConfigMap` entri alih-alih akses, Anda perlu memperbaruinya untuk mengakses metrik tambahan dari bidang kontrol cluster Amazon EKS. Untuk konfigurasi yang diperbarui, lihat [Mengkonfigurasi Amazon EKS secara manual untuk akses scraper](#) di Amazon Managed Service for Prometheus User Guide.

Langkah 2: Gunakan metrik Prometheus

Untuk informasi selengkapnya tentang cara menggunakan metrik Prometheus setelah Anda mengaktifkannya untuk klaster, lihat Panduan Pengguna Layanan [Terkelola Amazon](#) untuk Prometheus.

Langkah 3: Kelola pencakar Prometheus

Untuk mengelola pencakar, pilih tab Observability di konsol Amazon EKS. Tabel menunjukkan daftar pencakar untuk cluster, termasuk informasi seperti ID scraper, alias, status, dan tanggal pembuatan. Anda dapat menambahkan lebih banyak pencakar, mengedit pencakar, menghapus pencakar, atau melihat informasi lebih lanjut tentang pencakar saat ini.

Untuk melihat detail selengkapnya tentang scraper, pilih tautan ID scraper. Misalnya, Anda dapat melihat ARN, lingkungan, ID ruang kerja, peran IAM, konfigurasi, dan informasi jaringan. Anda dapat menggunakan ID scraper sebagai input ke Amazon Managed Service untuk operasi API Prometheus seperti, dan. [DescribeScraperUpdateScraperDeleteScraper](#) Untuk informasi selengkapnya tentang penggunaan API Prometheus, lihat Referensi API [Amazon Managed Service](#) for Prometheus.

Menyebarkan Prometheus menggunakan Helm

Sebagai alternatif untuk menggunakan Amazon Managed Service untuk Prometheus, Anda dapat menerapkan Prometheus ke cluster Anda dengan Helm. Jika Anda sudah menginstal Helm, Anda dapat memeriksa versi Anda dengan perintah `helm version`. Helm adalah manajer paket untuk kluster Kubernetes. Untuk informasi lebih lanjut tentang Helm dan cara menginstal layanan ini, lihat [the section called “Menerapkan aplikasi dengan Helm”](#).

Setelah Anda mengonfigurasi Helm untuk kluster Amazon EKS Anda, Anda dapat menggunakannya untuk men-deploy Prometheus dengan langkah-langkah berikut.

1. Buat namespace Prometheus.

```
kubectl create namespace prometheus
```

2. Tambahkan bagan repositori prometheus-community.

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
```

3. Terapkan Prometheus.

```
helm upgrade -i prometheus prometheus-community/prometheus \
  --namespace prometheus \
  --set alertmanager.persistence.storageClass="gp2" \
  --set server.persistentVolume.storageClass="gp2"
```

Note

Jika terdapat kesalahan `Error: failed to download "stable/prometheus"` (hint: running `helm repo update` may help) saat menjalankan perintah ini, jalankan `helm repo update prometheus-community`, dan kemudian coba jalankan perintah Langkah 2 lagi.

Jika terdapat kesalahan `Error: rendered manifests contain a resource that already exists`, jalankan `helm uninstall your-release-name -n namespace`, lalu coba jalankan kembali perintah Langkah 3.

4. Verifikasi bahwa semua Pod di `prometheus` namespace berada dalam status. `READY`

```
kubectl get pods -n prometheus
```

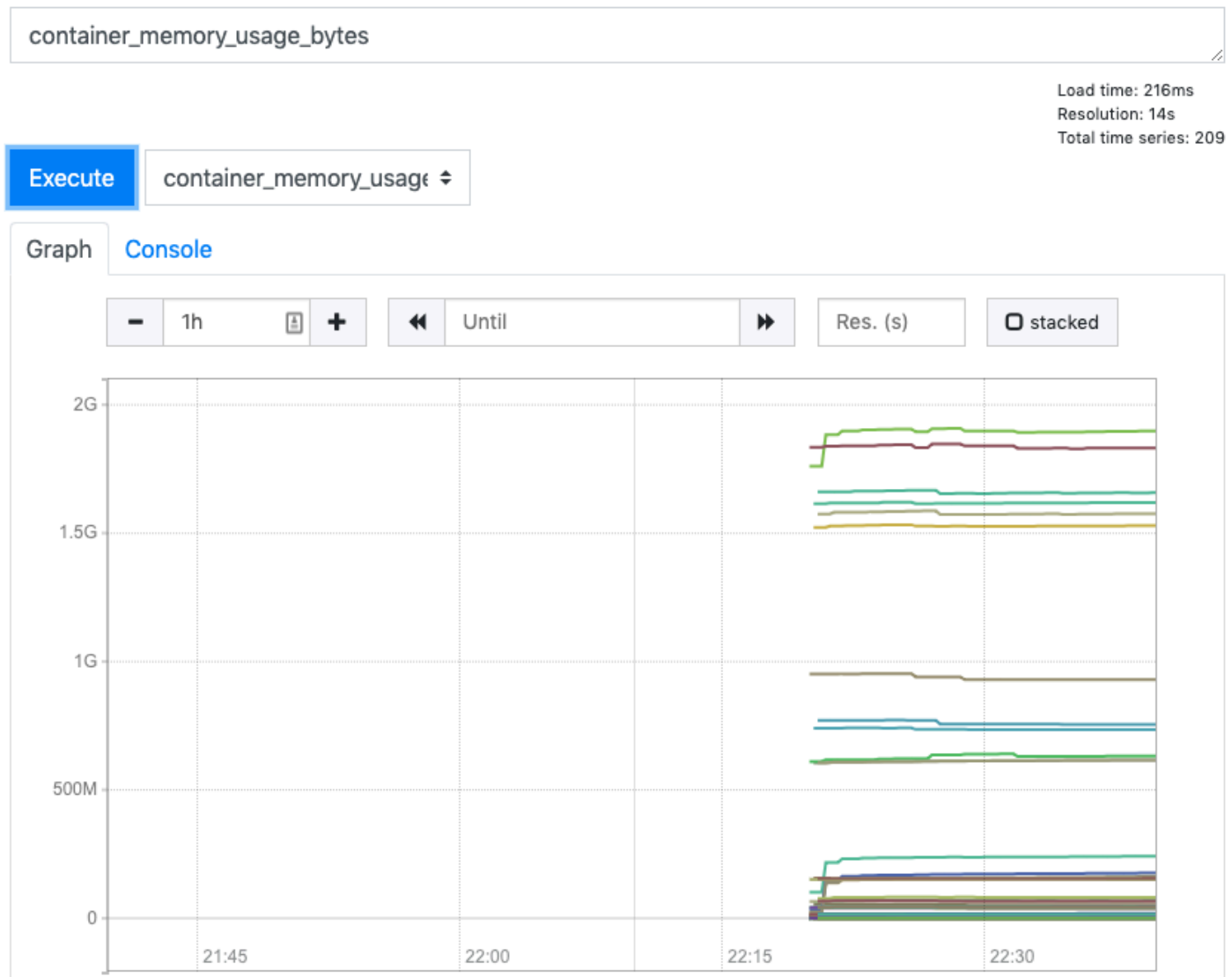
Contoh output adalah sebagai berikut.

NAME	READY	STATUS	RESTARTS	AGE
<code>prometheus-alertmanager-59b4c8c744-r7bgp</code>	1/2	Running	0	48s
<code>prometheus-kube-state-metrics-7cfd87cf99-jkz2f</code>	1/1	Running	0	48s
<code>prometheus-node-exporter-jcjzq</code>	1/1	Running	0	48s
<code>prometheus-node-exporter-jxv2h</code>	1/1	Running	0	48s
<code>prometheus-node-exporter-vbdks</code>	1/1	Running	0	48s
<code>prometheus-pushgateway-76c444b68c-82tnw</code>	1/1	Running	0	48s
<code>prometheus-server-775957f748-mmht9</code>	1/2	Running	0	48s

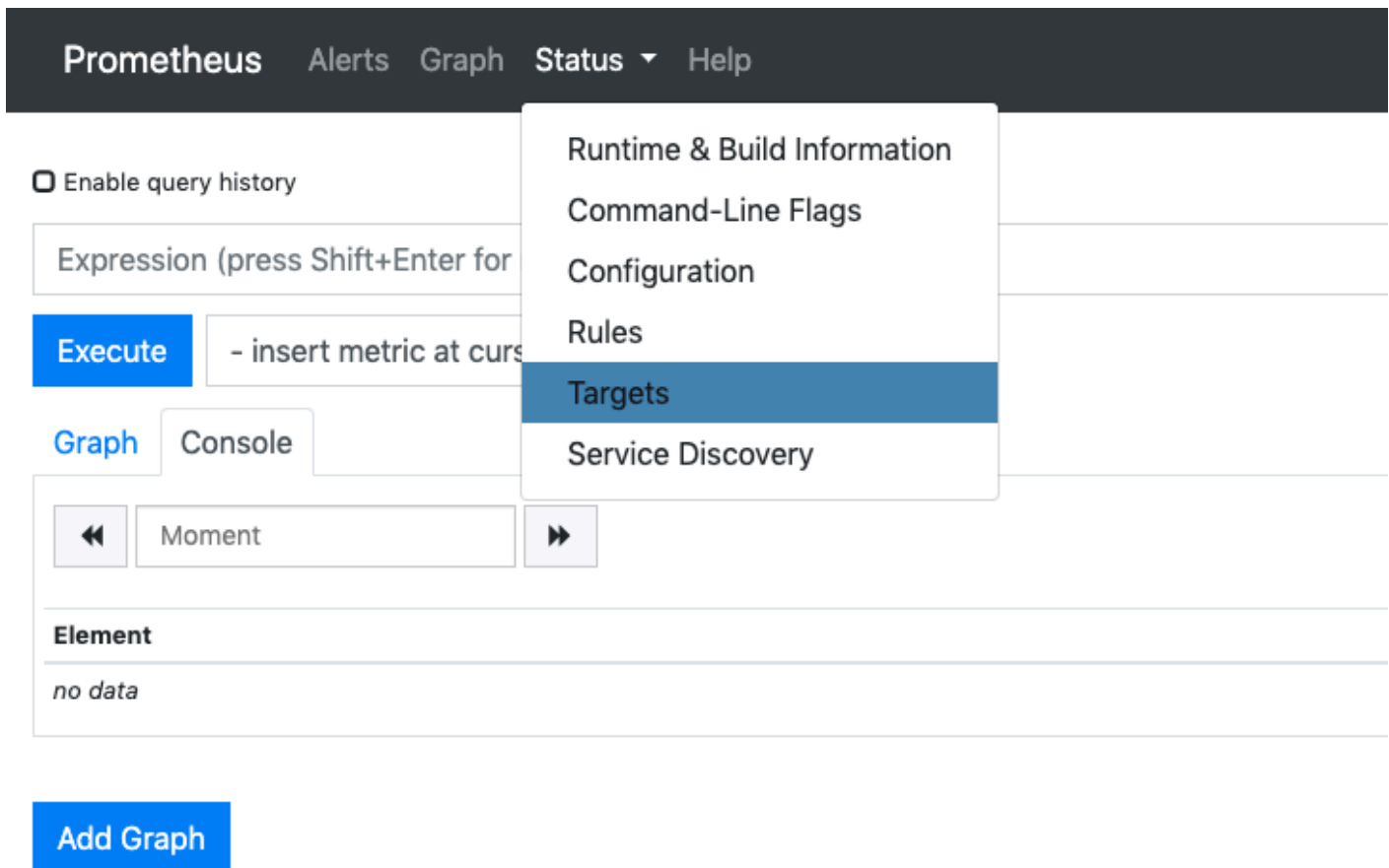
5. Gunakan `kubectl` untuk meneruskan port konsol Prometheus ke komputer lokal Anda.

```
kubectl --namespace=prometheus port-forward deploy/prometheus-server 9090
```

6. Arahkan browser web <http://localhost:9090> untuk melihat konsol Prometheus.
7. Pilih metrik dari menu - masukkan metrik pada kursor, lalu pilih Eksekusi. Pilih tabel Grafik untuk menampilkan metrik dari waktu ke waktu. Citra berikut menunjukkan `container_memory_usage_bytes` dari waktu ke waktu.



8. Dari bilah atas navigasi, pilih Status, kemudian Target.



Semua titik akhir Kubernetes yang terhubung ke Prometheus dengan menggunakan penemuan layanan sedang ditampilkan.

Ambil metrik mentah bidang kontrol dalam format Prometheus

[Bidang kontrol Kubernetes memperlihatkan sejumlah metrik yang direpresentasikan dalam format Prometheus.](#) Metrik ini berguna untuk pemantauan dan analisis. Mereka diekspos secara internal melalui titik akhir metrik, dan dapat diakses tanpa sepenuhnya menggunakan Prometheus. Namun, menerapkan Prometheus lebih mudah memungkinkan analisis metrik dari waktu ke waktu.

Untuk melihat output metrik mentah, ganti endpoint dan jalankan perintah berikut.

```
kubectl get --raw endpoint
```

Perintah ini memungkinkan Anda untuk melewati jalur titik akhir apa pun dan mengembalikan respons mentah. Output mencantumkan metrik yang berbeda line-by-line, dengan setiap baris termasuk nama metrik, tag, dan nilai.

```
metric_name{tag="value"[,...]} value
```

Ambil metrik dari server API

Titik akhir server API umum diekspos di bidang kontrol Amazon EKS. Titik akhir ini terutama berguna ketika melihat metrik tertentu.

```
kubectl get --raw /metrics
```

Contoh output adalah sebagai berikut.

```
[...]
# HELP rest_client_requests_total Number of HTTP requests, partitioned by status code,
method, and host.
# TYPE rest_client_requests_total counter
rest_client_requests_total{code="200",host="127.0.0.1:21362",method="POST"} 4994
rest_client_requests_total{code="200",host="127.0.0.1:443",method="DELETE"} 1
rest_client_requests_total{code="200",host="127.0.0.1:443",method="GET"} 1.326086e+06
rest_client_requests_total{code="200",host="127.0.0.1:443",method="PUT"} 862173
rest_client_requests_total{code="404",host="127.0.0.1:443",method="GET"} 2
rest_client_requests_total{code="409",host="127.0.0.1:443",method="POST"} 3
rest_client_requests_total{code="409",host="127.0.0.1:443",method="PUT"} 8
# HELP ssh_tunnel_open_count Counter of ssh tunnel total open attempts
# TYPE ssh_tunnel_open_count counter
ssh_tunnel_open_count 0
# HELP ssh_tunnel_open_fail_count Counter of ssh tunnel failed open attempts
# TYPE ssh_tunnel_open_fail_count counter
ssh_tunnel_open_fail_count 0
```

Output mentah ini mengembalikan kata per kata yang ditampilkan server API.

Ambil metrik bidang kontrol dengan **metrics.eks.amazonaws.com**

Untuk cluster yang versi Kubernetes ke 1.28 atas, Amazon EKS juga mengekspos metrik di bawah grup API **metrics.eks.amazonaws.com**. Metrik ini mencakup komponen bidang kontrol seperti kube-scheduler dan kube-controller-manager.

Note

Jika Anda memiliki konfigurasi webhook yang dapat memblokir pembuatan APIService sumber daya baru **v1.metrics.eks.amazonaws.com** di kluster Anda, fitur titik akhir metrik

mungkin tidak tersedia. Anda dapat memverifikasinya di log kube-apiserver audit dengan mencari `v1.metrics.eks.amazonaws.com` kata kunci.

Ambil metrik **kube-scheduler**

Untuk mengambil kube-scheduler metrik, gunakan perintah berikut.

```
kubectl get --raw "/apis/metrics.eks.amazonaws.com/v1/ksh/container/metrics"
```

Contoh output adalah sebagai berikut.

```
# TYPE scheduler_pending_pods gauge
scheduler_pending_pods{queue="active"} 0
scheduler_pending_pods{queue="backoff"} 0
scheduler_pending_pods{queue="gated"} 0
scheduler_pending_pods{queue="unschedulable"} 18
# HELP scheduler_pod_scheduling_attempts [STABLE] Number of attempts to successfully
  schedule a pod.
# TYPE scheduler_pod_scheduling_attempts histogram
scheduler_pod_scheduling_attempts_bucket{le="1"} 79
scheduler_pod_scheduling_attempts_bucket{le="2"} 79
scheduler_pod_scheduling_attempts_bucket{le="4"} 79
scheduler_pod_scheduling_attempts_bucket{le="8"} 79
scheduler_pod_scheduling_attempts_bucket{le="16"} 79
scheduler_pod_scheduling_attempts_bucket{le="+Inf"} 81
[...]
```

Ambil metrik **kube-controller-manager**

Untuk mengambil kube-controller-manager metrik, gunakan perintah berikut.

```
kubectl get --raw "/apis/metrics.eks.amazonaws.com/v1/kcm/container/metrics"
```

Contoh output adalah sebagai berikut.

```
[...]
workqueue_work_duration_seconds_sum{name="pvprotection"} 0
workqueue_work_duration_seconds_count{name="pvprotection"} 0
workqueue_work_duration_seconds_bucket{name="replicaset",le="1e-08"} 0
workqueue_work_duration_seconds_bucket{name="replicaset",le="1e-07"} 0
```

```

workqueue_work_duration_seconds_bucket{name="replicaset",le="1e-06"} 0
workqueue_work_duration_seconds_bucket{name="replicaset",le="9.999999999999999e-06"} 0
workqueue_work_duration_seconds_bucket{name="replicaset",le="9.999999999999999e-05"} 19
workqueue_work_duration_seconds_bucket{name="replicaset",le="0.001"} 109
workqueue_work_duration_seconds_bucket{name="replicaset",le="0.01"} 139
workqueue_work_duration_seconds_bucket{name="replicaset",le="0.1"} 181
workqueue_work_duration_seconds_bucket{name="replicaset",le="1"} 191
workqueue_work_duration_seconds_bucket{name="replicaset",le="10"} 191
workqueue_work_duration_seconds_bucket{name="replicaset",le="+Inf"} 191
workqueue_work_duration_seconds_sum{name="replicaset"} 4.265655885000002
[...]
```

Memahami metrik scheduler dan controller manager

Tabel berikut menjelaskan metrik penjadwal dan manajer pengontrol yang tersedia untuk pengikisan gaya Prometheus. Untuk informasi selengkapnya tentang metrik ini, lihat [Referensi Metrik Kubernetes di dokumentasi](#) Kubernetes.

Metrik	Komponen bidang kontrol	Deskripsi
<code>scheduler_pending_pods</code>	penjadwal	Jumlah Pod yang menunggu untuk dijadwalkan ke sebuah node untuk dieksekusi.
<code>scheduler_schedule_attempts_total</code>	penjadwal	Jumlah upaya yang dilakukan untuk menjadwalkan Pod.
<code>scheduler_preemption_attempts_total</code>	penjadwal	Jumlah upaya yang dilakukan oleh penjadwal untuk menjadwalkan Pod prioritas yang lebih tinggi dengan mengusir Pod prioritas yang lebih rendah.
<code>scheduler_preemption_victims</code>	penjadwal	Jumlah Pod yang telah dipilih untuk pengusuran untuk memberi ruang bagi Pod dengan prioritas yang lebih tinggi.

Metrik	Komponen bidang kontrol	Deskripsi
<code>scheduler_pod_scheduling_attempts</code>	penjadwal	Jumlah upaya untuk berhasil menjadwalkan sebuah Pod.
<code>scheduler_scheduling_attempt_duration_seconds</code>	penjadwal	Menunjukkan seberapa cepat atau lambat scheduler dapat menemukan tempat yang cocok untuk menjalankan Pod berdasarkan berbagai faktor seperti ketersediaan sumber daya dan aturan penjadwalan.
<code>scheduler_pod_scheduling_latency_duration_seconds</code>	penjadwal	end-to-end Latensi untuk Pod sedang dijadwalkan, sejak Pod memasuki antrean penjadwalan. Ini mungkin melibatkan beberapa upaya penjadwalan.
<code>cronjob_controller_job_creation_skew_duration_seconds</code>	manajer pengontrol	Waktu antara kapan cronjob dijadwalkan untuk dijalankan, dan saat pekerjaan yang sesuai dibuat.
<code>workqueue_depth</code>	manajer pengontrol	Kedalaman antrian saat ini.
<code>workqueue_adds_total</code>	manajer pengontrol	Jumlah total penambahan ditangani oleh workqueue.
<code>workqueue_queue_duration_seconds</code>	manajer pengontrol	Waktu dalam hitungan detik item tetap berada dalam antrean kerja sebelum diminta.
<code>workqueue_work_duration_seconds</code>	manajer pengontrol	Waktu dalam hitungan detik memproses item dari antrean kerja membutuhkan waktu.

Terapkan scraper Prometheus untuk mengikis metrik secara konsisten

Untuk menerapkan scraper Prometheus untuk mengikis metrik secara konsisten, gunakan konfigurasi berikut:

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: prometheus-conf
data:
  prometheus.yml: |-
    global:
      scrape_interval: 30s
    scrape_configs:
      # apiserver metrics
      - job_name: apiserver-metrics
        kubernetes_sd_configs:
          - role: endpoints
        scheme: https
        tls_config:
          ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
          insecure_skip_verify: true
        bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
        relabel_configs:
          - source_labels:
              [
                __meta_kubernetes_namespace,
                __meta_kubernetes_service_name,
                __meta_kubernetes_endpoint_port_name,
              ]
            action: keep
            regex: default;kubernetes;https
      # Scheduler metrics
      - job_name: 'ksh-metrics'
        kubernetes_sd_configs:
          - role: endpoints
        metrics_path: /apis/metrics.eks.amazonaws.com/v1/ksh/container/metrics
        scheme: https
        tls_config:
          ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
          insecure_skip_verify: true
        bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
```

```

    relabel_configs:
    - source_labels:
        [
            __meta_kubernetes_namespace,
            __meta_kubernetes_service_name,
            __meta_kubernetes_endpoint_port_name,
        ]
      action: keep
      regex: default;kubernetes;https
# Controller Manager metrics
- job_name: 'kcm-metrics'
  kubernetes_sd_configs:
  - role: endpoints
  metrics_path: /apis/metrics.eks.amazonaws.com/v1/kcm/container/metrics
  scheme: https
  tls_config:
    ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
    insecure_skip_verify: true
  bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
  relabel_configs:
  - source_labels:
      [
          __meta_kubernetes_namespace,
          __meta_kubernetes_service_name,
          __meta_kubernetes_endpoint_port_name,
      ]
    action: keep
    regex: default;kubernetes;https
---
apiVersion: v1
kind: Pod
metadata:
  name: prom-pod
spec:
  containers:
  - name: prom-container
    image: prom/prometheus
    ports:
    - containerPort: 9090
    volumeMounts:
    - name: config-volume
      mountPath: /etc/prometheus/
  volumes:
  - name: config-volume

```

```
configMap:
  name: prometheus-conf
```

Izin yang mengikuti diperlukan untuk Pod untuk mengakses endpoint metrik baru.

```
{
  "effect": "allow",
  "apiGroups": [
    "metrics.eks.amazonaws.com"
  ],
  "resources": [
    "kcm/metrics",
    "ksh/metrics"
  ],
  "verbs": [
    "get"
  ] },
```

Untuk menambal peran yang digunakan, Anda dapat menggunakan perintah berikut.

```
kubectl patch clusterrole <role-name> --type=json -p='[
  {
    "op": "add",
    "path": "/rules/-",
    "value": {
      "verbs": ["get"],
      "apiGroups": ["metrics.eks.amazonaws.com"],
      "resources": ["kcm/metrics", "ksh/metrics"]
    }
  }
]
```

Kemudian Anda dapat melihat dasbor Prometheus dengan mem-proxy port scraper Prometheus ke port lokal Anda.

```
kubectl port-forward pods/prom-pod 9090:9090
```

Untuk kluster Amazon EKS Anda, metrik bidang kontrol Kubernetes ini juga dimasukkan ke dalam Metrik Amazon CloudWatch di bawah namespace. AWS/EKS Untuk melihatnya, buka [CloudWatch konsol](#) dan pilih Semua metrik dari panel navigasi kiri. Pada halaman pemilihan Metrik, pilih AWS/EKS namespace dan dimensi metrik untuk kluster Anda.

Pantau data cluster dengan Amazon CloudWatch

Amazon CloudWatch adalah layanan pemantauan yang mengumpulkan metrik dan log dari sumber daya cloud Anda. CloudWatch menyediakan beberapa metrik Amazon EKS dasar secara gratis saat menggunakan cluster baru yang versi 1.28 ke atas. Namun, saat menggunakan Operator CloudWatch Observabilitas sebagai add-on Amazon EKS, Anda dapat memperoleh fitur observabilitas yang disempurnakan.

Metrik dasar di Amazon CloudWatch

Untuk cluster yang versi 1.28 Kubernetes ke atas, Anda mendapatkan metrik CloudWatch vendored secara gratis di namespace. AWS/EKS Tabel berikut memberikan daftar metrik dasar yang tersedia untuk versi yang didukung. Setiap metrik yang terdaftar memiliki frekuensi satu menit.

Nama metrik	Deskripsi
<code>scheduler_schedule_attempts_total</code>	Jumlah total upaya oleh scheduler untuk menjadwalkan Pod dalam kluster selama periode tertentu. Metrik ini membantu memantau beban kerja penjadwal dan dapat menunjukkan tekanan penjadwalan atau potensi masalah dengan penempatan Pod. Unit: Hitung Statistik yang valid: Jumlah
<code>scheduler_schedule_attempts_SCHEDULED</code>	Jumlah upaya yang berhasil oleh scheduler untuk menjadwalkan Pod ke node di cluster selama periode tertentu. Unit: Hitung Statistik yang valid: Jumlah
<code>scheduler_schedule_attempts_UNSCHEDULABLE</code>	Jumlah upaya untuk menjadwalkan Pod yang tidak dapat dijadwalkan untuk periode tertentu karena kendala yang valid, seperti CPU atau

Nama metrik	Deskripsi
	memori yang tidak mencukupi pada sebuah node. Unit: Hitung Statistik yang valid: Jumlah
<code>scheduler_schedule_attempts_ERROR</code>	Jumlah upaya untuk menjadwalkan Pod yang gagal untuk periode tertentu karena masalah internal dengan scheduler itu sendiri, seperti masalah konektivitas API Server. Unit: Hitung Statistik yang valid: Jumlah
<code>scheduler_pending_pods</code>	Jumlah total Pod yang tertunda yang akan dijadwalkan oleh scheduler di klaster selama periode tertentu. Unit: Hitung Statistik yang valid: Jumlah
<code>scheduler_pending_pods_ACTIVEQ</code>	Jumlah Pod yang tertunda di ActiveQ, yang menunggu untuk dijadwalkan di cluster untuk periode tertentu. Unit: Hitung Statistik yang valid: Jumlah

Nama metrik	Deskripsi
<code>scheduler_pending_pods_UNSCHEDULABLE</code>	Jumlah Pod yang tertunda yang coba dijadwalkan dan gagal oleh scheduler, dan disimpan dalam keadaan tidak terjadwal untuk dicoba lagi. Unit: Hitung Statistik yang valid: Jumlah
<code>scheduler_pending_pods_BACKOFF</code>	Jumlah Pod yang tertunda backoffQ dalam status backoff yang menunggu periode backoff mereka berakhir. Unit: Hitung Statistik yang valid: Jumlah
<code>scheduler_pending_pods_GATED</code>	Jumlah Pod yang tertunda yang saat ini menunggu dalam keadaan terjaga keamanannya karena tidak dapat dijadwalkan sampai memenuhi persyaratan yang diperlukan. Unit: Hitung Statistik yang valid: Jumlah
<code>apiserver_request_total</code>	Jumlah permintaan HTTP yang dibuat di semua server API di cluster. Unit: Hitung Statistik yang valid: Jumlah

Nama metrik	Deskripsi
<code>apiserver_request_total_4XX</code>	Jumlah permintaan HTTP yang dibuat ke semua server API di cluster yang menghasilkan kode status 4XX (kesalahan klien). Unit: Hitung Statistik yang valid: Jumlah
<code>apiserver_request_total_429</code>	Jumlah permintaan HTTP yang dibuat untuk semua server API di cluster yang menghasilkan kode 429 status, yang terjadi ketika klien melebihi ambang batas batas tingkat. Unit: Hitung Statistik yang valid: Jumlah
<code>apiserver_request_total_5XX</code>	Jumlah permintaan HTTP yang dibuat untuk semua server API di cluster yang menghasilkan kode status 5XX (kesalahan server). Unit: Hitung Statistik yang valid: Jumlah
<code>apiserver_request_total_LIST_PODS</code>	Jumlah permintaan LIST Pod yang dibuat untuk semua server API di cluster. Unit: Hitung Statistik yang valid: Jumlah

Nama metrik	Deskripsi
<code>apiserver_request_duration_seconds_PUT_P99</code>	Persentil ke-99 latensi untuk PUT permintaan dihitung dari semua permintaan di semua server API di cluster. Merupakan waktu respons di bawah mana 99% dari semua PUT permintaan diselesaikan. Unit: Detik Statistik yang valid: Rata-rata
<code>apiserver_request_duration_seconds_PATCH_P99</code>	Persentil ke-99 latensi untuk PATCH permintaan dihitung dari semua permintaan di semua server API di cluster. Merupakan waktu respons di bawah mana 99% dari semua PATCH permintaan diselesaikan. Unit: Detik Statistik yang valid: Rata-rata
<code>apiserver_request_duration_seconds_POST_P99</code>	Persentil ke-99 latensi untuk POST permintaan dihitung dari semua permintaan di semua server API di cluster. Merupakan waktu respons di bawah mana 99% dari semua POST permintaan diselesaikan. Unit: Detik Statistik yang valid: Rata-rata

Nama metrik	Deskripsi
<code>apiserver_request_duration_seconds_GET_P99</code>	Persentil ke-99 latensi untuk GET permintaan dihitung dari semua permintaan di semua server API di cluster. Merupakan waktu respons di bawah mana 99% dari semua GET permintaan diselesaikan. Unit: Detik Statistik yang valid: Rata-rata
<code>apiserver_request_duration_seconds_LIST_P99</code>	Persentil ke-99 latensi untuk LIST permintaan dihitung dari semua permintaan di semua server API di cluster. Merupakan waktu respons di bawah mana 99% dari semua LIST permintaan diselesaikan. Unit: Detik Statistik yang valid: Rata-rata
<code>apiserver_request_duration_seconds_DELETE_P99</code>	Persentil ke-99 latensi untuk DELETE permintaan dihitung dari semua permintaan di semua server API di cluster. Merupakan waktu respons di bawah mana 99% dari semua DELETE permintaan diselesaikan. Unit: Detik Statistik yang valid: Rata-rata

Nama metrik	Deskripsi
<code>apiserver_current_inflight_requests_MUTATING</code>	Jumlah permintaan mutasi (POST,, PUTDELETE,PATCH) yang saat ini sedang diproses di semua server API di cluster. Metrik ini mewakili permintaan yang sedang dalam penerbangan dan belum menyelesaikan pemrosesan. Unit: Hitung Statistik yang valid: Jumlah
<code>apiserver_current_inflight_requests_READONLY</code>	Jumlah permintaan hanya-baca (GET,LIST) yang saat ini sedang diproses di semua server API di cluster. Metrik ini mewakili permintaan yang sedang dalam penerbangan dan belum menyelesaikan pemrosesan. Unit: Hitung Statistik yang valid: Jumlah
<code>apiserver_admission_webhook_request_total</code>	Jumlah permintaan webhook masuk yang dibuat di semua server API di cluster. Unit: Hitung Statistik yang valid: Jumlah
<code>apiserver_admission_webhook_request_total_ADMIT</code>	Jumlah permintaan webhook masuk yang bermutasi yang dibuat di semua server API di cluster. Unit: Hitung Statistik yang valid: Jumlah

Nama metrik	Deskripsi
<code>apiserver_admission_webhook_request_total_VALIDATING</code>	Jumlah permintaan webhook masuk yang memvalidasi yang dibuat di semua server API di cluster. Unit: Hitung Statistik yang valid: Jumlah
<code>apiserver_admission_webhook_rejection_count</code>	Jumlah permintaan webhook masuk yang dibuat di semua server API di cluster yang ditolak. Unit: Hitung Statistik yang valid: Jumlah
<code>apiserver_admission_webhook_rejection_count_ADMIT</code>	Jumlah permintaan webhook penerimaan yang bermutasi yang dibuat di semua server API di cluster yang ditolak. Unit: Hitung Statistik yang valid: Jumlah
<code>apiserver_admission_webhook_rejection_count_VALIDATING</code>	Jumlah permintaan webhook masuk yang memvalidasi yang dibuat di semua server API di cluster yang ditolak. Unit: Hitung Statistik yang valid: Jumlah

Nama metrik	Deskripsi
<code>apiserver_admission_webhook_admission_duration_seconds</code>	Persentil ke-99 latensi untuk permintaan webhook penerimaan pihak ketiga dihitung dari semua permintaan di semua server API di cluster. Merupakan waktu respons di bawah mana 99% dari semua permintaan webhook penerimaan pihak ketiga diselesaikan. Unit: Detik Statistik yang valid: Rata-rata
<code>apiserver_admission_webhook_admission_duration_seconds_ADMIT_P99</code>	Persentil ke-99 latensi untuk permintaan webhook penerimaan bermutasi pihak ketiga dihitung dari semua permintaan di semua server API di cluster. Merupakan waktu respons di bawah mana 99% dari semua permintaan webhook penerimaan bermutasi pihak ketiga diselesaikan. Unit: Detik Statistik yang valid: Rata-rata
<code>apiserver_admission_webhook_admission_duration_seconds_VALIDATING_P99</code>	Persentil ke-99 latensi untuk permintaan webhook penerimaan validasi pihak ketiga dihitung dari semua permintaan di semua server API di cluster. Merupakan waktu respons di bawah mana 99% dari semua permintaan webhook penerimaan validasi pihak ketiga diselesaikan. Unit: Detik Statistik yang valid: Rata-rata

Nama metrik	Deskripsi
<code>apiserver_storage_size_bytes</code>	Ukuran fisik dalam byte file database penyimpanan etcd yang digunakan oleh server API di cluster. Metrik ini mewakili ruang disk aktual yang dialokasikan untuk penyimpanan. Unit: Bytes Statistik yang valid: Maksimum

Operator CloudWatch Observabilitas Amazon

Amazon CloudWatch Observability mengumpulkan log, metrik, dan data penelusuran waktu nyata. Ini mengirim mereka ke [Amazon CloudWatch](#) dan [AWS X-Ray](#). Anda dapat menginstal add-on ini untuk mengaktifkan Sinyal CloudWatch Aplikasi dan CloudWatch Wawasan Kontainer dengan peningkatan observabilitas untuk Amazon EKS. Ini membantu Anda memantau kesehatan dan kinerja infrastruktur dan aplikasi kontainer Anda. Operator CloudWatch Observabilitas Amazon dirancang untuk menginstal dan mengonfigurasi komponen yang diperlukan.

Amazon EKS mendukung Operator CloudWatch Observabilitas sebagai [add-on Amazon EKS](#). Add-on ini memungkinkan Container Insights pada node pekerja Linux dan Windows di cluster. Untuk mengaktifkan Wawasan Kontainer di Windows, versi add-on Amazon EKS harus 1.5.0 atau lebih tinggi. Saat ini, Sinyal CloudWatch Aplikasi tidak didukung di Amazon EKS Windows.

Topik di bawah ini menjelaskan cara memulai menggunakan CloudWatch Observability Operator untuk kluster Amazon EKS Anda.

- Untuk petunjuk cara menginstal add-on ini, lihat [Menginstal CloudWatch agen dengan add-on Amazon CloudWatch Observability EKS atau bagan Helm di Panduan Pengguna](#) Amazon CloudWatch.
- Untuk informasi selengkapnya tentang Sinyal CloudWatch [Aplikasi](#), lihat [Sinyal Aplikasi](#) di Panduan CloudWatch Pengguna Amazon.
- Untuk informasi selengkapnya tentang Wawasan Kontainer, lihat [Menggunakan Wawasan Kontainer](#) di CloudWatch Panduan Pengguna Amazon.

Kirim log pesawat kontrol ke CloudWatch Log

Pencatatan pesawat kontrol Amazon EKS menyediakan log audit dan diagnostik langsung dari bidang kontrol Amazon EKS ke CloudWatch Log di akun Anda. Log ini memudahkan Anda untuk mengamankan dan menjalankan kluster Anda. Anda dapat memilih jenis log yang tepat yang Anda butuhkan, dan log dikirim sebagai aliran log ke grup untuk setiap kluster Amazon EKS. CloudWatch Anda dapat menggunakan filter CloudWatch langganan untuk melakukan analisis waktu nyata pada log atau meneruskannya ke layanan lain (log akan dikodekan dan dikompresi dengan format gzip Base64). Untuk informasi selengkapnya, lihat [CloudWatch pencatatan Amazon](#).

Anda dapat mulai menggunakan pencatatan bidang kendali Amazon EKS dengan memilih jenis log mana yang ingin Anda aktifkan untuk setiap kluster Amazon EKS yang baru atau yang sudah ada. Anda dapat mengaktifkan atau menonaktifkan setiap jenis log pada basis per-cluster menggunakan, AWS Management Console AWS CLI (versi 1.16.139 atau lebih tinggi), atau melalui Amazon EKS API. Saat diaktifkan, log secara otomatis dikirim dari kluster Amazon EKS ke CloudWatch Log di akun yang sama.

Saat Anda menggunakan pencatatan pesawat kontrol Amazon EKS, Anda dikenakan harga Amazon EKS standar untuk setiap cluster yang Anda jalankan. Anda akan dikenakan biaya konsumsi data CloudWatch Log standar dan biaya penyimpanan untuk setiap log yang dikirim ke CloudWatch Log dari cluster Anda. Anda juga dikenakan biaya untuk AWS sumber daya apa pun, seperti EC2 instans Amazon atau volume Amazon EBS, yang Anda sediakan sebagai bagian dari kluster Anda.

Jenis log bidang kendali kluster berikut tersedia. Setiap jenis log sesuai dengan komponen bidang kendali Kubernetes. Untuk mempelajari selengkapnya tentang komponen ini, lihat [Komponen-Komponen Kubernetes](#) dalam dokumentasi Kubernetes.

Server API (**api**)

Server API kluster Anda adalah komponen control plane yang mengekspos Kubernetes API. Jika Anda mengaktifkan log server API saat meluncurkan kluster, atau segera setelahnya, log tersebut menyertakan flag server API yang digunakan untuk memulai server API. Untuk informasi selengkapnya, lihat [kube-apiserver](#) dan [kebijakan audit](#) dalam dokumentasi Kubernetes.

Audit (**audit**)

Log audit Kubernetes menyediakan catatan pengguna individu, administrator, atau komponen sistem yang telah memengaruhi kluster Anda. Untuk informasi selengkapnya, lihat [Auditing](#) dalam dokumentasi Kubernetes.

Authenticator () **authenticator**

Log Authenticator unik untuk Amazon EKS. Log-log ini mewakili komponen bidang kendali yang digunakan Amazon EKS untuk autentikasi Kubernetes [Role Based Access Control](#) (RBAC) menggunakan kredensial IAM. Untuk informasi selengkapnya, lihat [Manajemen klaster](#).

Manajer pengontrol (**controllerManager**)

Manajer pengontrol mengelola loop kontrol inti yang dikirimkan bersama Kubernetes. Untuk informasi selengkapnya, lihat [kube-controller-manager](#) dalam dokumentasi Kubernetes.

Penjadwal () **scheduler**

Komponen scheduler mengelola kapan dan di mana menjalankan Pod di klaster Anda. Untuk informasi selengkapnya, lihat [penjadwal-kube](#) dalam dokumentasi Kubernetes.

Mengaktifkan atau menonaktifkan log bidang kontrol

Secara default, log bidang kontrol cluster tidak dikirim ke CloudWatch Log. Anda harus mengaktifkan setiap jenis log satu per satu untuk mengirim log untuk klaster Anda. CloudWatch Tingkat konsumsi log, penyimpanan arsip, dan pemindaian data berlaku untuk log bidang kontrol yang diaktifkan. Untuk informasi selengkapnya, lihat [harga CloudWatch](#).

Untuk memperbarui konfigurasi pencatatan bidang kontrol, Amazon EKS memerlukan hingga lima alamat IP yang tersedia di setiap subnet. Saat Anda mengaktifkan jenis log, maka log tersebut akan dikirim dengan tingkat verbositas log sebesar 2.

Anda dapat mengaktifkan atau menonaktifkan log bidang kontrol dengan [AWS CLI](#) [AWS Management Console](#) atau.

AWS Management Console

1. Buka [konsol Amazon EKS](#).
2. Pilih nama klaster untuk menampilkan informasi klaster Anda.
3. Pilih tab Observability.
4. Di bagian Control plane logging, pilih Manage logging.
5. Untuk setiap jenis log individu, pilih apakah jenis log harus dihidupkan atau dimatikan. Secara default, setiap jenis log dimatikan.

6. Pilih Simpan perubahan untuk menyelesaikan.

AWS CLI

1. Periksa versi AWS CLI Anda dengan perintah berikut.

```
aws --version
```

Jika versi AWS CLI Anda lebih awal dari 1.16.139, Anda harus terlebih dahulu memperbarui ke versi terbaru. Untuk menginstal atau memutakhirkan AWS CLI, lihat [Menginstal Antarmuka Baris AWS Perintah di Panduan Pengguna Antarmuka Baris AWS Perintah](#).

2. Perbarui konfigurasi ekspor log bidang kontrol cluster Anda dengan perintah AWS CLI berikut. Ganti *my-cluster* dengan nama cluster Anda dan tentukan nilai akses titik akhir yang Anda inginkan.

Note

Perintah berikut mengirimkan semua jenis log yang tersedia ke CloudWatch Log.

```
aws eks update-cluster-config \
  --region region-code \
  --name my-cluster \
  --logging '{"clusterLogging":[{"types":
["api","audit","authenticator","controllerManager","scheduler"],"enabled":true}]}'
```

Contoh output adalah sebagai berikut.

```
{
  "update": {
    "id": "883405c8-65c6-4758-8cee-2a7c1340a6d9",
    "status": "InProgress",
    "type": "LoggingUpdate",
    "params": [
      {
        "type": "ClusterLogging",
        "value": "{\"clusterLogging\":{\"types\":[\"api\", \"audit\",
\"authenticator\", \"controllerManager\", \"scheduler\"], \"enabled\":true}}"
```

```

    ],
    "createdAt": 1553271814.684,
    "errors": []
  }
}

```

3. Pantau status pembaruan konfigurasi log Anda dengan perintah berikut, menggunakan nama kluster dan ID pembaruan yang dikembalikan oleh perintah sebelumnya. Pembaruan Anda selesai saat status muncul sebagai berikut `Successful`.

```

aws eks describe-update \
  --region region-code \
  --name my-cluster \
  --update-id 883405c8-65c6-4758-8cee-2a7c1340a6d9

```

Contoh output adalah sebagai berikut.

```

{
  "update": {
    "id": "883405c8-65c6-4758-8cee-2a7c1340a6d9",
    "status": "Successful",
    "type": "LoggingUpdate",
    "params": [
      {
        "type": "ClusterLogging",
        "value": "{\"clusterLogging\": [{\"types\": [\"api\", \"audit\", \"authenticator\", \"controllerManager\", \"scheduler\"], \"enabled\": true}]}"
      }
    ],
    "createdAt": 1553271814.684,
    "errors": []
  }
}

```

Lihat log bidang kontrol kluster

Setelah Anda mengaktifkan salah satu jenis log bidang kontrol untuk kluster Amazon EKS Anda, Anda dapat melihatnya di CloudWatch konsol.

Untuk mempelajari selengkapnya tentang melihat, menganalisis, dan mengelola log in CloudWatch, lihat [Panduan Pengguna CloudWatch Log Amazon](#).

1. Buka [konsol CloudWatch](#) . Tautan dapat membuka konsol dan menampilkan grup log yang tersedia saat ini, serta memfilternya dengan prefiks /aws/eks.
2. Pilih klaster yang ingin Anda lihat log-nya. Format nama grup log adalah /aws/eks/*my-cluster*/cluster.
3. Pilih aliran log untuk melihat. Daftar berikut mendeskripsikan format nama aliran log untuk setiap jenis log.

 Note

Saat data aliran log bertambah, nama aliran log dirotasikan. Ketika beberapa aliran log ada untuk jenis log tertentu, Anda dapat melihat aliran log terbaru dengan mencari nama aliran log dengan waktu peristiwa Terakhir terbaru.

- Log komponen server Kubernetes API (**api**) – kube-apiserver-*1234567890abcdef01234567890abcde*
 - Audit (**audit**) – kube-apiserver-audit-*1234567890abcdef01234567890abcde*
 - Authenticator (**authenticator**) – authenticator-*1234567890abcdef01234567890abcde*
 - Manajer pengendali (**controllerManager**) – kube-controller-manager-*1234567890abcdef01234567890abcde*
 - Penjadwal (**scheduler**) – kube-scheduler-*1234567890abcdef01234567890abcde*
4. Lihat melalui peristiwa aliran log.

Misalnya, Anda akan melihat flag server API awal untuk cluster saat melihat bagian atas. kube-apiserver-*1234567890abcdef01234567890abcde*

 Note

Jika Anda tidak melihat log server API di awal aliran log, kemungkinan file log server API diputar di server sebelum Anda mengaktifkan log server API di server. File log apa pun yang diputar sebelum pencatatan server API diaktifkan tidak dapat diekspor ke CloudWatch

Namun, Anda dapat membuat kluster baru dengan versi Kubernetes yang sama dan mengaktifkan pencatatan server API ketika Anda membuat kluster. Cluster dengan versi platform yang sama mengaktifkan flag yang sama, jadi flag Anda harus cocok dengan flag cluster baru. Ketika Anda selesai melihat flag untuk cluster baru di CloudWatch, Anda dapat menghapus cluster baru.

Log panggilan API sebagai AWS CloudTrail peristiwa

Amazon EKS terintegrasi dengan AWS CloudTrail. CloudTrail adalah layanan yang menyediakan catatan tindakan oleh pengguna, peran, atau AWS layanan di Amazon EKS. CloudTrail menangkap semua panggilan API untuk Amazon EKS sebagai acara. Ini termasuk panggilan dari konsol Amazon EKS dan dari panggilan kode ke operasi Amazon EKS API.

Jika Anda membuat jejak, Anda dapat mengaktifkan pengiriman CloudTrail acara secara terus menerus ke bucket Amazon S3. Ini termasuk acara untuk Amazon EKS. Jika Anda tidak mengonfigurasi jejak, Anda masih dapat melihat peristiwa terbaru di CloudTrail konsol dalam Riwayat acara. Menggunakan informasi yang CloudTrail dikumpulkan, Anda dapat menentukan beberapa detail tentang permintaan. Misalnya, Anda dapat menentukan kapan permintaan dibuat ke Amazon EKS, alamat IP tempat permintaan dibuat, dan siapa yang mengajukan permintaan.

Untuk mempelajari selengkapnya CloudTrail, lihat [Panduan AWS CloudTrail Pengguna](#).

Topik

- [Lihat referensi bermanfaat untuk AWS CloudTrail](#)
- [Menganalisis entri file AWS CloudTrail log](#)
- [Lihat metrik untuk grup EC2 Auto Scaling Amazon](#)

Lihat referensi bermanfaat untuk AWS CloudTrail

Saat Anda membuat AWS akun, juga CloudTrail diaktifkan di AWS akun Anda. Ketika aktivitas apa pun terjadi di Amazon EKS, aktivitas tersebut direkam dalam suatu CloudTrail peristiwa bersama dengan peristiwa AWS layanan lainnya dalam riwayat Acara. Anda dapat melihat, mencari, dan mengunduh acara terbaru di AWS akun Anda. Untuk informasi selengkapnya, lihat [Melihat peristiwa dengan riwayat CloudTrail acara](#).

Untuk catatan peristiwa yang sedang berlangsung di AWS akun Anda, termasuk acara untuk Amazon EKS, buat jejak. Jejak memungkinkan CloudTrail untuk mengirimkan file log ke bucket Amazon S3. Secara default, saat Anda membuat jejak di konsol, jejak tersebut berlaku untuk semua AWS Wilayah. Jejak mencatat peristiwa dari semua AWS Wilayah di AWS partisi dan mengirimkan file

log ke bucket Amazon S3 yang Anda tentukan. Selain itu, Anda dapat mengonfigurasi AWS layanan lain untuk menganalisis lebih lanjut dan menindaklanjuti data peristiwa yang dikumpulkan dalam CloudTrail log. Untuk informasi selengkapnya, lihat sumber daya berikut ini.

- [Ikhtisar untuk membuat jejak](#)
- [CloudTrail layanan dan integrasi yang didukung](#)
- [Mengonfigurasi notifikasi Amazon SNS untuk CloudTrail](#)
- [Menerima file CloudTrail log dari beberapa wilayah](#) dan [Menerima file CloudTrail log dari beberapa akun](#)

Semua tindakan Amazon EKS dicatat oleh CloudTrail dan didokumentasikan dalam [Referensi API Amazon EKS](#). Misalnya, panggilan ke [CreateCluster](#), [ListClusters](#) dan [DeleteCluster](#) bagian menghasilkan entri dalam file CloudTrail log.

Setiap peristiwa atau entri log berisi informasi tentang jenis identitas IAM yang membuat permintaan, dan kredensi mana yang digunakan. Jika kredensial sementara digunakan, entri menunjukkan bagaimana kredensialnya diperoleh.

Untuk informasi selengkapnya, lihat [Elemen userIdentity CloudTrail](#).

Menganalisis entri file AWS CloudTrail log

Trail adalah konfigurasi yang memungkinkan pengiriman peristiwa sebagai file log ke bucket Amazon S3 yang Anda tentukan. CloudTrail file log berisi satu atau lebih entri log. Peristiwa mewakili satu permintaan dari sumber manapun dan mencakup informasi tentang tindakan yang diminta. Ini termasuk informasi seperti tanggal dan waktu tindakan dan parameter permintaan yang digunakan. CloudTrail file log bukanlah jejak tumpukan yang diurutkan dari panggilan API publik, jadi file tersebut tidak muncul dalam urutan tertentu.

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan [CreateCluster](#) tindakan.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AKIAIOSFODNN7EXAMPLE",
    "arn": "arn:aws:iam::111122223333:user/username",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
```

```

    "userName": "username"
  },
  "eventTime": "2018-05-28T19:16:43Z",
  "eventSource": "eks.amazonaws.com",
  "eventName": "CreateCluster",
  "awsRegion": "region-code",
  "sourceIPAddress": "205.251.233.178",
  "userAgent": "PostmanRuntime/6.4.0",
  "requestParameters": {
    "resourcesVpcConfig": {
      "subnetIds": [
        "subnet-a670c2df",
        "subnet-4f8c5004"
      ]
    },
    "roleArn": "arn:aws: iam::111122223333:role/AWSServiceRoleForAmazonEKS-
CAC1G1VH3ZKZ",
    "clusterName": "test"
  },
  "responseElements": {
    "cluster": {
      "clusterName": "test",
      "status": "CREATING",
      "createdAt": 1527535003.208,
      "certificateAuthority": {},
      "arn": "arn:aws: eks:region-code:111122223333:cluster/test",
      "roleArn": "arn:aws: iam::111122223333:role/AWSServiceRoleForAmazonEKS-
CAC1G1VH3ZKZ",
      "version": "1.10",
      "resourcesVpcConfig": {
        "securityGroupIds": [],
        "vpcId": "vpc-21277358",
        "subnetIds": [
          "subnet-a670c2df",
          "subnet-4f8c5004"
        ]
      }
    }
  },
  "requestID": "a7a0735d-62ab-11e8-9f79-81ce5b2b7d37",
  "eventID": "eab22523-174a-499c-9dd6-91e7be3ff8e3",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"

```



```
}

```

Log entri untuk Peran Tertaut Layanan Amazon EKS

Peran terkait layanan Amazon EKS membuat panggilan API ke AWS sumber daya.

CloudTrail entri log dengan username: `AWSServiceRoleForAmazonEKS` dan username: `AWSServiceRoleForAmazonEKSNodegroup` muncul untuk panggilan yang dibuat oleh peran terkait layanan Amazon EKS. Untuk informasi selengkapnya tentang Amazon EKS dan peran tertaut layanan, lihat [the section called “Peran terkait layanan”](#).

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan

[DeleteInstanceProfile](#) tindakan yang dibuat oleh peran terkait

`AWSServiceRoleForAmazonEKSNodegroup` layanan, yang dicatat dalam `sessionContext`.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "ARO0A3WHGPEZ7SJ2CW55C5:EKS",
    "arn": "arn:aws: sts::111122223333:assumed-role/
AWSServiceRoleForAmazonEKSNodegroup/EKS",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "ARO0A3WHGPEZ7SJ2CW55C5",
        "arn": "arn:aws: iam::111122223333:role/aws-service-role/eks-
nodegroup.amazonaws.com/AWSServiceRoleForAmazonEKSNodegroup",
        "accountId": "111122223333",
        "userName": "AWSServiceRoleForAmazonEKSNodegroup"
      },
      "webIdFederationData": {},
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2020-02-26T00:56:33Z"
      }
    },
    "invokedBy": "eks-nodegroup.amazonaws.com"
  },
  "eventTime": "2020-02-26T00:56:34Z",
  "eventSource": "iam.amazonaws.com",

```

```
{
  "eventName": "DeleteInstanceProfile",
  "awsRegion": "region-code",
  "sourceIPAddress": "eks-nodegroup.amazonaws.com",
  "userAgent": "eks-nodegroup.amazonaws.com",
  "requestParameters": {
    "instanceProfileName": "eks-11111111-2222-3333-4444-abcdef123456"
  },
  "responseElements": null,
  "requestID": "11111111-2222-3333-4444-abcdef123456",
  "eventID": "11111111-2222-3333-4444-abcdef123456",
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
```

Lihat metrik untuk grup EC2 Auto Scaling Amazon

Grup node terkelola Amazon EKS mengaktifkan metrik grup EC2 Auto Scaling Amazon secara default tanpa biaya tambahan. Grup Auto Scaling mengirimkan data sampel ke Amazon CloudWatch setiap menit. Metrik ini dapat disempurnakan dengan nama grup Auto Scaling. Mereka memberi Anda visibilitas berkelanjutan ke dalam riwayat grup Auto Scaling yang memberi daya pada grup node terkelola Anda, seperti perubahan ukuran grup dari waktu ke waktu. Metrik grup Auto Scaling tersedia di [Amazon](#) atau konsol Auto CloudWatch Scaling. Untuk informasi selengkapnya, lihat [Memantau CloudWatch metrik untuk grup dan instans Auto Scaling Anda](#).

Dengan kumpulan metrik grup Auto Scaling, Anda dapat memantau penskalaan grup node terkelola. Metrik grup Auto Scaling melaporkan ukuran minimum, maksimum, dan yang diinginkan dari grup Auto Scaling. Anda dapat membuat alarm jika jumlah node dalam grup node berada di bawah ukuran minimum, yang akan menunjukkan grup node yang tidak sehat. Melacak ukuran grup node juga berguna dalam menyesuaikan jumlah maksimum sehingga bidang data Anda tidak kehabisan kapasitas.

Jika Anda memilih untuk tidak mengumpulkan metrik ini, Anda dapat memilih untuk menonaktifkan semua atau hanya beberapa dari mereka. Misalnya, Anda dapat melakukan ini untuk menghindari kebisingan di CloudWatch dasbor Anda. Untuk informasi selengkapnya, lihat [CloudWatch metrik Amazon untuk EC2 Auto Scaling Amazon](#).

Kirim metrik dan lacak data dengan ADOT Operator

Amazon EKS mendukung penggunaan AWS Management Console, AWS CLI dan Amazon EKS API untuk menginstal dan mengelola [AWS Distro for OpenTelemetry \(ADOT\)](#) Operator. [Ini memudahkan](#)

[aplikasi Anda yang berjalan di Amazon EKS untuk mengirim metrik dan melacak data ke beberapa opsi layanan pemantauan seperti Amazon CloudWatch, Prometheus, dan X-Ray.](#)

Untuk informasi selengkapnya, lihat [Memulai dengan AWS Distro untuk OpenTelemetry menggunakan Eks Add-On](#) di AWS Distro untuk dokumentasi. OpenTelemetry

Melihat data agregat tentang sumber daya kluster dengan Dasbor EKS

image::images/eks-dashboard.png[screenshot of account level cluster metrics]

Apa itu Dasbor Amazon EKS?

Dasbor Amazon EKS menyediakan visibilitas terkonsolidasi ke dalam kluster Kubernetes Anda di beberapa Wilayah dan Akun. AWS AWS Dengan dasbor ini, Anda dapat:

- Lacak cluster yang dijadwalkan untuk end-of-support upgrade otomatis dalam 90 hari ke depan.
- Proyek EKS mengontrol biaya pesawat untuk cluster dalam dukungan yang diperpanjang.
- Tinjau cluster dengan wawasan yang perlu diperhatikan sebelum memutakhirkan.
- Identifikasi grup node terkelola yang menjalankan versi AMI tertentu.
- Memantau distribusi tipe dukungan cluster (standar dibandingkan dengan dukungan yang diperluas).

Dasbor EKS terintegrasi dengan EKS Cluster Insights untuk memunculkan masalah dengan cluster Anda, seperti penggunaan Kubernetes yang sudah usang. APIs Untuk informasi selengkapnya, lihat [the section called “Wawasan cluster”](#).

Note

Dasbor EKS tidak real-time dan diperbarui setiap 12 jam. Untuk pemantauan kluster waktu nyata, lihat [Monitor cluster](#)

Bagaimana dasbor menggunakan AWS Organizations?

Dasbor Amazon EKS memerlukan integrasi AWS Organizations untuk fungsionalitas. Ini memanfaatkan AWS Organizations untuk mengumpulkan informasi cluster dengan aman di seluruh

akun. Integrasi ini menyediakan manajemen dan tata kelola terpusat sebagai skala AWS infrastruktur Anda.

Jika AWS Organizations tidak diaktifkan untuk infrastruktur Anda, lihat [Panduan Pengguna AWS Organizations](#) untuk petunjuk penyiapan.

Akses lintas wilayah dan lintas akun

Dasbor EKS dapat melihat sumber daya cluster di akun apa pun yang merupakan anggota AWS organisasi. Untuk membuat daftar AWS akun di organisasi Anda, lihat [Mengeksplor detail untuk semua akun di organisasi](#).

Wilayah AWS us-east-1 menghasilkan dasbor. Anda harus masuk ke wilayah ini untuk melihat dasbor. Dasbor mengumpulkan data di seluruh AWS wilayah, tetapi ini tidak termasuk GovCloud atau wilayah Tiongkok.

Istilah kunci

- **AWS Organisasi:** Struktur manajemen terpadu untuk beberapa AWS akun.
- **Akun manajemen:** Akun utama yang mengontrol AWS Organisasi.
- **Akun anggota:** Akun apa pun dalam organisasi kecuali akun manajemen.
- **Administrator yang didelegasikan:** Akun anggota memberikan izin administratif lintas akun tertentu. Dalam akun manajemen, Anda dapat memilih satu akun administrator yang didelegasikan per AWS Layanan.
- **Akses tepercaya:** Otorisasi untuk Dasbor EKS untuk mengakses informasi klaster di seluruh akun organisasi.
- **Service-Linked Role (SLRs):** Jenis unik peran IAM yang ditautkan langsung ke layanan. AWS Dasbor EKS menggunakan SLR untuk membaca informasi tentang akun dan organisasi Anda.
- Untuk informasi selengkapnya, lihat [Terminologi dan konsep](#) di Panduan Pengguna AWS Organizations.

Ikhtisar umum

1. Akses akun manajemen AWS Organisasi Anda.

- Langkah-langkah untuk mengakses akun manajemen bergantung pada cara Anda mengonfigurasi AWS Organisasi Anda. Misalnya, Anda dapat mengakses akun manajemen melalui [Pusat AWS Identitas](#) atau [Okta](#).

2. Aktifkan akses Tepercaya melalui Konsol EKS.
3. Tetapkan administrator Delegasi menggunakan ID AWS Akun mereka.
4. Beralih ke akun administrator yang didelegasikan.
5. Akses Konsol EKS yang disempurnakan dengan visibilitas seluruh organisasi.

Aktifkan Dasbor EKS menggunakan AWS konsol

Important

Anda harus masuk ke Akun Manajemen AWS Organisasi Anda untuk mengaktifkan Dasbor EKS.

Akses pengaturan Dasbor EKS

1. Konfirmasi hal berikut:
 - a. Anda memiliki AWS Organizations diaktifkan dan dikonfigurasi.
 - b. Anda masuk ke akun Manajemen organisasi.
 - c. Anda sedang melihat Konsol AWS Manajemen di wilayah us-east-1.
2. Arahkan ke konsol EKS.
3. Di bilah sisi kiri, buka Pengaturan Dasbor.

Siapkan akses ke Dasbor Amazon EKS

1. Temukan ID AWS Akun AWS Akun yang ingin Anda izinkan untuk melihat Dasbor EKS.
 - a. Langkah ini opsional, tetapi disarankan. Jika tidak, Anda hanya dapat mengakses dasbor dari akun manajemen. Sebagai praktik terbaik, Anda harus membatasi akses ke akun manajemen.
2. Klik Aktifkan akses tepercaya.
 - a. Anda sekarang dapat melihat dasbor dari akun manajemen.
3. Klik Daftarkan administrator yang didelegasikan dan masukkan ID Akun AWS Akun yang akan Anda gunakan untuk melihat dasbor.
 - a. Anda sekarang dapat melihat dasbor dari akun administrator yang didelegasikan atau akun manajemen.

Untuk informasi tentang izin yang diperlukan untuk mengaktifkan dasbor, lihat [Kebijakan IAM minimum](#) yang diperlukan.

Lihat dasbor EKS

1. Masuk ke akun administrator yang didelegasikan (disarankan) atau akun manajemen.
2. Masuk ke wilayah us-east-1.
3. Buka layanan EKS, dan pilih Dasbor dari bilah sisi kiri.

Note

[Tinjau izin IAM yang diperlukan untuk melihat dasbor EKS.](#)

Konfigurasi dasbor

Anda dapat mengonfigurasi tampilan dasbor, dan memfilter sumber daya.

Sumber daya yang tersedia

- Cluster: Lihat informasi agregat tentang status dan lokasi Kluster EKS.
 - Cluster dengan masalah kesehatan.
 - Cluster pada EKS Extended Support.
 - Rincian cluster menurut versi Kubernetes.
- Grup node terkelola: Tinjau grup dan EC2 Instans simpul Terkelola.
 - Grup node menurut tipe AMI, seperti Amazon Linux atau Bottlerocket.
 - Masalah kesehatan kelompok simpul.
 - Distribusi tipe instans.
- Pengaya: Pelajari tentang Add-on Amazon EKS yang telah Anda instal, dan statusnya.
 - Jumlah instalasi per add-on.
 - Add-on dengan masalah kesehatan.
 - Distribusi versi per add-on.

Tampilan yang tersedia

- Tampilan grafik
 - Tampilan widget yang dapat disesuaikan menampilkan grafik dan visualisasi sumber daya yang dipilih.
 - Perubahan pada tampilan Grafik, seperti menghapus widget, terlihat oleh semua pengguna Dasbor EKS.
- Tampilan sumber daya
 - Tampilan daftar sumber daya yang dipilih, mendukung filter.
- Tampilan Peta
 - Lihat distribusi geografis dari sumber daya yang dipilih.

Filter dasbor EKS

Anda dapat memfilter Dasbor EKS dengan:

- AWS Akun
- Unit organisasi, didefinisikan oleh AWS Organizations
- AWS Wilayah

Nonaktifkan dasbor EKS menggunakan AWS konsol

1. Konfirmasi hal berikut:
 - a. Anda memiliki AWS Organizations diaktifkan dan dikonfigurasi.
 - b. Anda masuk ke akun Manajemen organisasi.
 - c. Anda sedang melihat Konsol AWS Manajemen di wilayah us-east-1.
2. Arahkan ke konsol EKS.
3. Di bilah sisi kiri, buka Pengaturan Dasbor.
4. Klik Nonaktifkan akses tepercaya.

Memecahkan masalah dasbor EKS

Masalah mengaktifkan dasbor EKS

- Anda harus masuk ke akun manajemen AWS Organisasi.
- Jika Anda tidak memiliki AWS Organisasi, buatlah satu. Pelajari cara [Membuat dan mengonfigurasi organisasi](#).
- Jika AWS akun Anda sudah menjadi anggota AWS Organisasi, identifikasi administrator organisasi.
- Anda harus masuk ke AWS akun dengan izin IAM yang memadai untuk membuat dan memperbarui sumber daya AWS Organizations.

Masalah melihat dasbor EKS

- Anda harus masuk ke salah satu AWS akun berikut:
 - Akun manajemen AWS Organisasi
 - Akun administrator yang didelegasikan, diidentifikasi dalam pengaturan dasbor EKS dari akun manajemen.
- Jika Anda baru saja mengaktifkan Dasbor EKS, harap perhatikan bahwa populasi data awal dapat memakan waktu hingga 12 jam.
- [Coba aktifkan kembali dasbor menggunakan CLI](#), termasuk membuat peran terkait layanan.

Widget dasbor bergerak secara tak terduga

- Dasbor EKS menyimpan tampilan widget yang dapat dikonfigurasi di tingkat AWS Akun. Jika Anda mengubah tampilan widget, orang lain yang menggunakan AWS akun yang sama akan melihat perubahannya.

Konfigurasikan integrasi Dasbor EKS dengan AWS Organizations

Bagian ini memberikan step-by-step petunjuk untuk mengonfigurasi integrasi Dasbor EKS dengan AWS Organizations. Anda akan mempelajari cara mengaktifkan dan menonaktifkan akses tepercaya antar layanan, serta cara mendaftar dan membatalkan pendaftaran akun administrator yang didelegasikan. Setiap tugas konfigurasi dapat dilakukan menggunakan AWS konsol atau AWS CLI.

Aktifkan akses terpercaya

Akses terpercaya mengizinkan Dasbor EKS untuk mengakses informasi kluster dengan aman di semua akun di organisasi Anda.

Menggunakan AWS konsol

1. Masuk ke akun manajemen AWS Organisasi Anda.
2. Arahkan ke konsol EKS di wilayah us-east-1.
3. Di bilah sisi kiri, pilih Pengaturan Dasbor.
4. Klik Aktifkan akses terpercaya.

Note

Saat Anda mengaktifkan akses terpercaya melalui konsol EKS, sistem secara otomatis membuat peran `AWSServiceRoleForAmazonEKSDashboard` terkait layanan. Pembuatan otomatis ini tidak terjadi jika Anda mengaktifkan akses terpercaya menggunakan konsol AWS CLI atau AWS Organizations.

Menggunakan AWS CLI

1. Masuk ke akun manajemen AWS Organisasi Anda.
2. Jalankan perintah berikut:

```
aws iam create-service-linked-role --aws-service-name dashboard.eks.amazonaws.com
aws organizations enable-aws-service-access --service-principal eks.amazonaws.com
```

Nonaktifkan akses terpercaya

Menonaktifkan akses terpercaya akan mencabut izin Dasbor EKS untuk mengakses informasi kluster di seluruh akun organisasi Anda.

Menggunakan AWS konsol

1. Masuk ke akun manajemen AWS Organisasi Anda.
2. Arahkan ke Konsol EKS di wilayah us-east-1.

3. Di bilah sisi kiri, pilih Pengaturan Dasbor.
4. Klik Nonaktifkan akses tepercaya.

Menggunakan AWS CLI

1. Masuk ke akun manajemen AWS Organisasi Anda.
2. Jalankan perintah berikut:

```
aws organizations disable-aws-service-access --service-principal eks.amazonaws.com
```

Aaktifkan akun administrator yang didelegasikan

Administrator yang didelegasikan adalah akun anggota yang diberikan izin untuk mengakses Dasbor EKS.

Menggunakan AWS konsol

1. Masuk ke akun manajemen AWS Organisasi Anda.
2. Arahkan ke konsol EKS di wilayah us-east-1.
3. Di bilah sisi kiri, pilih Pengaturan Dasbor.
4. Klik Daftarkan administrator yang didelegasikan.
5. Masukkan ID Akun AWS Akun yang ingin Anda pilih sebagai administrator yang didelegasikan.
6. Konfirmasikan pendaftaran.

Menggunakan AWS CLI

1. Masuk ke akun manajemen AWS Organisasi Anda.
2. Jalankan perintah berikut, ganti 123456789012 dengan ID akun Anda:

```
aws organizations register-delegated-administrator --account-id 123456789012 --  
service-principal eks.amazonaws.com
```

Nonaktifkan akun administrator yang didelegasikan

Menonaktifkan administrator yang didelegasikan akan menghapus izin akun untuk mengakses Dasbor EKS.

Menggunakan AWS konsol

1. Masuk ke akun manajemen AWS Organisasi Anda.
2. Arahkan ke konsol EKS di wilayah us-east-1.
3. Di bilah sisi kiri, pilih Pengaturan Dasbor.
4. Temukan administrator yang didelegasikan dalam daftar.
5. Klik Deregister di samping akun yang ingin Anda hapus sebagai administrator yang didelegasikan.

Menggunakan AWS CLI

1. Masuk ke akun manajemen AWS Organisasi Anda.
2. Jalankan perintah berikut, ganti 123456789012 dengan ID akun administrator yang didelegasikan:

```
aws organizations deregister-delegated-administrator --account-id 123456789012 --  
service-principal eks.amazonaws.com
```

Diperlukan kebijakan IAM minimum

Bagian ini menguraikan kebijakan IAM minimum yang diperlukan untuk mengaktifkan akses tepercaya dan mendelegasikan administrator untuk integrasi Dasbor EKS dengan Organizations. AWS

Kebijakan untuk mengaktifkan akses tepercaya

Untuk mengaktifkan akses tepercaya antara Dasbor EKS dan AWS Organizations, Anda memerlukan izin berikut:

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",
```

```

        "Action": [
            "organizations:EnableAWSServiceAccess",
            "organizations:DescribeOrganization",
            "organizations:ListAWSServiceAccessForOrganization"
        ],
        "Resource": "*"
    },
    {
        "Effect": "Allow",
        "Action": [
            "iam:CreateServiceLinkedRole"
        ],
        "Resource": "arn:aws:iam::*:role/aws-service-role/
dashboard.eks.amazonaws.com/AWSServiceRoleForAmazonEKSDashboard"
    }
]
}

```

Kebijakan untuk mendelegasikan administrator

Untuk mendaftarkan atau membatalkan pendaftaran administrator yang didelegasikan untuk Dasbor EKS, Anda memerlukan izin berikut:

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "organizations:RegisterDelegatedAdministrator",
                "organizations:DeregisterDelegatedAdministrator",
                "organizations:ListDelegatedAdministrators"
            ],
            "Resource": "*"
        }
    ]
}

```

Kebijakan untuk melihat Dasbor EKS

```

{
    "Version": "2012-10-17",
    "Statement": [

```

```

    {
      "Sid": "AmazonEKSDashboardReadOnly",
      "Effect": "Allow",
      "Action": [
        "eks:ListDashboardData",
        "eks:ListDashboardResources",
        "eks:DescribeClusterVersions"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AmazonOrganizationsReadOnly",
      "Effect": "Allow",
      "Action": [
        "organizations:DescribeOrganization",
        "organizations:ListAWSServiceAccessForOrganization",
        "organizations:ListRoots",
        "organizations:ListAccountsForParent",
        "organizations:ListOrganizationalUnitsForParent"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AmazonOrganizationsDelegatedAdmin",
      "Effect": "Allow",
      "Action": [
        "organizations:ListDelegatedAdministrators"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "organizations:ServicePrincipal": "eks.amazonaws.com"
        }
      }
    }
  ]
}

```

 Note

Kebijakan ini harus dilampirkan pada prinsipal IAM (pengguna atau peran) dalam akun manajemen AWS Organisasi Anda. Akun anggota tidak dapat mengaktifkan akses tepercaya atau mendelegasikan administrator.

Tingkatkan EKS dengan AWS layanan terintegrasi

Selain layanan yang tercakup dalam bagian lain, Amazon EKS bekerja dengan lebih banyak AWS layanan untuk memberikan solusi tambahan. Topik ini mengidentifikasi beberapa layanan lain yang menggunakan Amazon EKS untuk menambahkan fungsionalitas, atau layanan yang digunakan Amazon EKS untuk melakukan tugas.

Topik

- [Buat sumber daya Amazon EKS dengan AWS CloudFormation](#)
- [Analisis peristiwa keamanan di EKS dengan Amazon Detective](#)
- [Mendeteksi ancaman dengan Amazon GuardDuty](#)
- [Menilai ketahanan kluster EKS dengan Resilience Hub AWS](#)
- [Memusatkan dan menganalisis data keamanan EKS dengan Security Lake](#)
- [Aktifkan konektivitas lintas cluster yang aman dengan Amazon VPC Lattice](#)
- [Luncurkan kluster EKS latensi rendah dengan Local Zones AWS](#)

Buat sumber daya Amazon EKS dengan AWS CloudFormation

Amazon EKS terintegrasi dengan AWS CloudFormation, layanan yang membantu Anda memodelkan dan mengatur AWS sumber daya Anda sehingga Anda dapat menghabiskan lebih sedikit waktu untuk membuat dan mengelola sumber daya dan infrastruktur Anda. Anda membuat template yang menjelaskan semua AWS sumber daya yang Anda inginkan, misalnya kluster Amazon EKS, dan AWS CloudFormation menangani penyediaan dan konfigurasi sumber daya tersebut untuk Anda.

Bila Anda menggunakan AWS CloudFormation, Anda dapat menggunakan kembali template Anda untuk mengatur sumber daya Amazon EKS Anda secara konsisten dan berulang kali. Cukup jelaskan sumber daya Anda sekali, lalu sediakan sumber daya yang sama berulang kali di beberapa AWS akun dan Wilayah.

Amazon EKS dan AWS CloudFormation template

Untuk menyediakan dan mengonfigurasi sumber daya untuk Amazon EKS dan layanan terkait, Anda harus memahami [templat AWS CloudFormation](#). Templat adalah file teks yang diformat dalam JSON atau YAML. Template ini menjelaskan sumber daya yang ingin Anda sediakan di AWS CloudFormation tumpukan Anda. Jika Anda tidak terbiasa dengan JSON atau YAMAL, Anda dapat

menggunakan AWS CloudFormation Designer untuk membantu Anda memulai dengan template. AWS CloudFormation Untuk informasi lebih lanjut, lihat [Apa itu AWS CloudFormation Desainer?](#) dalam AWS CloudFormation User Guide.

Amazon EKS mendukung menciptakan kluster dan grup simpul di AWS CloudFormation. Untuk informasi selengkapnya, termasuk contoh templat JSON dan YAMAL untuk sumber daya Amazon EKS Anda, lihat [referensi jenis sumber daya Amazon EKS](#) di AWS CloudFormation Panduan Pengguna.

Pelajari lebih lanjut tentang AWS CloudFormation

Untuk mempelajari selengkapnya AWS CloudFormation, lihat sumber daya berikut:

- [AWS CloudFormation](#)
- [AWS CloudFormation Panduan Pengguna](#)
- [AWS CloudFormation Panduan Pengguna Antarmuka Baris Perintah](#)

Analisis peristiwa keamanan di EKS dengan Amazon Detective

[Amazon Detective](#) membantu Anda menganalisis, menyelidiki, dan mengidentifikasi akar penyebab temuan keamanan atau aktivitas mencurigakan dengan cepat. Detective secara otomatis mengumpulkan data log dari sumber daya Anda. AWS kemudian menggunakan pembelajaran mesin, analisis statistik, dan teori grafik untuk menghasilkan visualisasi yang membantu Anda melakukan penyelidikan keamanan yang lebih cepat dan lebih efisien. Agregasi data Detective prebuilt, ringkasan, dan konteks membantu Anda menganalisis dan menentukan sifat dan tingkat kemungkinan masalah keamanan dengan cepat. Untuk informasi selengkapnya, lihat [Panduan Pengguna Detektif Amazon](#).

Detective mengatur Kubernetes dan data ke dalam temuan-temuan seperti: AWS

- Detail kluster Amazon EKS, termasuk identitas IAM yang menciptakan cluster dan peran layanan cluster. Anda dapat menyelidiki aktivitas API AWS dan Kubernetes dari identitas IAM ini dengan Detective.
- Detail kontainer, seperti konteks gambar dan keamanan. Anda juga dapat meninjau detail untuk Pod yang dihentikan.
- Aktivitas API Kubernetes, termasuk tren keseluruhan dalam aktivitas API dan detail panggilan API tertentu. Misalnya, Anda dapat menunjukkan jumlah panggilan API Kubernetes yang berhasil dan

gagal yang dikeluarkan selama rentang waktu yang dipilih. Selain itu, bagian tentang panggilan API yang baru diamati mungkin berguna untuk mengidentifikasi aktivitas yang mencurigakan.

Log audit Amazon EKS adalah paket sumber data opsional yang dapat ditambahkan ke grafik perilaku Detektif Anda. Anda dapat melihat paket sumber opsional yang tersedia, dan statusnya di akun Anda. Untuk informasi selengkapnya, lihat [log audit Amazon EKS untuk Detektif](#) di Panduan Pengguna Detektif Amazon.

Gunakan Amazon Detective dengan Amazon EKS

Sebelum Anda dapat meninjau temuan, Detektif harus diaktifkan setidaknya selama 48 jam di AWS Wilayah yang sama dengan kluster Anda. Untuk informasi selengkapnya, lihat [Menyiapkan Detektif Amazon](#) di Panduan Pengguna Detektif Amazon.

1. Buka konsol Detektif di <https://console.aws.amazon.com/detective/>
2. Dari panel navigasi kiri, pilih Cari.
3. Pilih Jenis dan kemudian pilih EKS cluster.
4. Masukkan nama cluster atau ARN lalu pilih Cari.
5. Di hasil penelusuran, pilih nama kluster yang ingin Anda lihat aktivitasnya. Untuk informasi selengkapnya tentang apa yang dapat Anda lihat, lihat [Keseluruhan aktivitas API Kubernetes yang melibatkan kluster Amazon EKS](#) di Panduan Pengguna Detektif Amazon.

Mendeteksi ancaman dengan Amazon GuardDuty

Amazon GuardDuty adalah layanan deteksi ancaman yang membantu melindungi akun, wadah, beban kerja, dan data Anda dengan AWS lingkungan Anda. Menggunakan model machine learning (ML), serta kemampuan deteksi anomali dan ancaman, GuardDuty terus memantau berbagai sumber log dan aktivitas runtime untuk mengidentifikasi dan memprioritaskan potensi risiko keamanan dan aktivitas berbahaya di lingkungan Anda.

Di antara fitur-fitur lainnya, GuardDuty menawarkan dua fitur berikut yang mendeteksi potensi ancaman terhadap kluster EKS Anda: Perlindungan EKS dan Pemantauan Runtime.

Note

Baru: Amazon EKS Auto Mode terintegrasi dengan GuardDuty.

Perlindungan EKS

Fitur ini menyediakan cakupan deteksi ancaman untuk membantu Anda melindungi kluster Amazon EKS dengan memantau log audit Kubernetes terkait. Log audit Kubernetes menangkap tindakan sekuensial dalam kluster Anda, termasuk aktivitas dari pengguna, aplikasi yang menggunakan Kubernetes API, dan control plane. Misalnya, GuardDuty dapat mengidentifikasi bahwa panggilan untuk berpotensi merusak sumber daya di kluster Kubernetes yang APIs dipanggil oleh pengguna yang tidak diautentikasi.

Saat Anda mengaktifkan Perlindungan EKS, GuardDuty akan dapat mengakses log audit Amazon EKS Anda hanya untuk deteksi ancaman berkelanjutan. Jika GuardDuty mengidentifikasi potensi ancaman terhadap kluster Anda, itu akan menghasilkan temuan log audit Kubernetes terkait dari tipe tertentu. Untuk informasi selengkapnya tentang jenis temuan yang tersedia dari log audit Kubernetes, lihat [jenis pencarian log audit Kubernetes di](#) Panduan Pengguna Amazon. GuardDuty

Untuk informasi selengkapnya, lihat [Perlindungan EKS](#) di Panduan GuardDuty Pengguna Amazon.

Pemantauan Runtime

Fitur ini memantau dan menganalisis tingkat sistem operasi, jaringan, dan peristiwa file untuk membantu Anda mendeteksi potensi ancaman dalam AWS beban kerja tertentu di lingkungan Anda.

Saat Anda mengaktifkan Runtime Monitoring dan menginstal GuardDuty agen di kluster Amazon EKS, GuardDuty mulai memantau peristiwa runtime yang terkait dengan cluster ini. Perhatikan bahwa GuardDuty agen dan Runtime Monitoring tidak tersedia untuk Amazon EKS Hybrid Nodes, jadi Runtime Monitoring tidak tersedia untuk peristiwa runtime yang terjadi pada node hybrid Anda. Jika GuardDuty mengidentifikasi potensi ancaman terhadap kluster Anda, itu akan menghasilkan temuan Runtime Monitoring terkait. Misalnya, ancaman berpotensi dimulai dengan mengorbankan satu wadah yang menjalankan aplikasi web yang rentan. Aplikasi web ini mungkin memiliki izin akses ke wadah dan beban kerja yang mendasarinya. Dalam skenario ini, kredensi yang tidak dikonfigurasi dengan benar berpotensi menyebabkan akses yang lebih luas ke akun, dan data yang tersimpan di dalamnya.

Untuk mengonfigurasi Runtime Monitoring, Anda menginstal GuardDuty agen ke kluster Anda sebagai add-on Amazon EKS. Untuk informasi selengkapnya add-on, lihat [the section called “AWS add-on”](#).

Untuk informasi selengkapnya, lihat [Runtime Monitoring](#) di Panduan GuardDuty Pengguna Amazon.

Menilai ketahanan kluster EKS dengan Resilience Hub AWS

AWS Resilience Hub menilai ketahanan kluster Amazon EKS dengan menganalisis infrastrukturnya. AWS Resilience Hub menggunakan konfigurasi kontrol akses berbasis peran Kubernetes (RBAC) untuk menilai beban kerja Kubernetes yang diterapkan ke kluster Anda. Untuk informasi selengkapnya, lihat [Mengaktifkan akses AWS Resilience Hub ke kluster Amazon EKS Anda di Panduan Pengguna](#) AWS Resilience Hub.

Memusatkan dan menganalisis data keamanan EKS dengan Security Lake

Amazon Security Lake adalah layanan danau data keamanan yang dikelola sepenuhnya yang memungkinkan Anda memusatkan data keamanan dari berbagai sumber, termasuk Amazon EKS. Dengan mengintegrasikan Amazon EKS dengan Security Lake, Anda dapat memperoleh wawasan yang lebih dalam tentang aktivitas yang dilakukan pada sumber daya Kubernetes Anda dan meningkatkan postur keamanan kluster Amazon EKS Anda.

Note

Untuk informasi selengkapnya tentang penggunaan Security Lake dengan Amazon EKS dan menyiapkan sumber data, lihat [dokumentasi Amazon Security Lake](#).

Manfaat menggunakan Security Lake dengan Amazon Amazon EKS

Data keamanan terpusat — Security Lake secara otomatis mengumpulkan dan memusatkan data keamanan dari kluster Amazon EKS Anda, bersama dengan data dari layanan lain, penyedia SaaS AWS, sumber lokal, dan sumber pihak ketiga. Ini memberikan pandangan komprehensif tentang postur keamanan Anda di seluruh organisasi Anda.

Format data standar — Security Lake mengubah data yang dikumpulkan ke dalam format [Open Cybersecurity Schema Framework \(OCSF\)](#), yang merupakan skema open-source standar. Normalisasi ini memungkinkan analisis dan integrasi yang lebih mudah dengan alat dan layanan keamanan lainnya.

Deteksi ancaman yang ditingkatkan — Dengan menganalisis data keamanan terpusat, termasuk log pesawat kontrol Amazon EKS, Anda dapat mendeteksi aktivitas yang berpotensi mencurigakan

dalam kluster Amazon EKS Anda dengan lebih efektif. Ini membantu dalam mengidentifikasi dan menanggapi insiden keamanan dengan segera.

Manajemen data yang disederhanakan — Security Lake mengelola siklus hidup data keamanan Anda dengan pengaturan retensi dan replikasi yang dapat disesuaikan. Ini menyederhanakan tugas manajemen data dan memastikan bahwa Anda menyimpan data yang diperlukan untuk tujuan kepatuhan dan audit.

Mengaktifkan Danau Keamanan untuk Amazon EKS

1. Aktifkan pencatatan pesawat kontrol Amazon EKS untuk kluster EKS Anda. Lihat [Mengaktifkan dan menonaktifkan log bidang kontrol](#) untuk instruksi terperinci.
2. [Tambahkan Amazon EKS Audit Log sebagai sumber di Security Lake](#). Security Lake kemudian akan mulai mengumpulkan informasi mendalam tentang aktivitas yang dilakukan pada sumber daya Kubernetes yang berjalan di kluster EKS Anda.
3. [Konfigurasi pengaturan retensi dan replikasi](#) untuk data keamanan Anda di Security Lake berdasarkan kebutuhan Anda.
4. Gunakan data OCSF yang dinormalisasi yang disimpan di Security Lake untuk respons insiden, analitik keamanan, dan integrasi dengan AWS layanan lain atau alat pihak ketiga. Misalnya, Anda dapat [Menghasilkan wawasan keamanan dari data Amazon Security Lake menggunakan Amazon OpenSearch Ingestion](#).

Menganalisis Log EKS di Danau Keamanan

Security Lake menormalkan peristiwa log EKS ke format OCSF, membuatnya lebih mudah untuk menganalisis dan menghubungkan data dengan peristiwa keamanan lainnya. Anda dapat menggunakan berbagai alat dan layanan, seperti Amazon Athena, Amazon, atau alat analisis keamanan pihak ketiga QuickSight, untuk menanyakan dan memvisualisasikan data yang dinormalisasi.

Untuk informasi lebih lanjut tentang pemetaan OCSF untuk peristiwa log EKS, lihat Log Audit <https://github.com/ocsf/examples/tree/main/mappings/markdown/> AWS/v1.1.0/EKS [referensi pemetaan] di repositori OCSF. GitHub

Aktifkan konektivitas lintas cluster yang aman dengan Amazon VPC Lattice

Amazon VPC Lattice adalah layanan jaringan aplikasi yang dikelola sepenuhnya yang dibangun langsung ke infrastruktur AWS jaringan yang dapat Anda gunakan untuk menghubungkan, mengamankan, dan memantau layanan Anda di beberapa akun dan Virtual Private Clouds (). VPCs [Dengan Amazon EKS, Anda dapat memanfaatkan Amazon VPC Lattice melalui penggunaan AWS Gateway API Controller, sebuah implementasi dari Kubernetes Gateway API](#). Menggunakan Amazon VPC Lattice, Anda dapat mengatur konektivitas lintas cluster dengan semantik Kubernetes standar dengan cara yang sederhana dan konsisten. Untuk mulai menggunakan Amazon VPC Lattice dengan Amazon EKS, lihat Panduan Pengguna [Pengontrol API AWS Gateway](#).

Luncurkan klaster EKS latensi rendah dengan Local Zones AWS

[Zona AWS Lokal](#) adalah perpanjangan dari AWS Wilayah dalam kedekatan geografis dengan pengguna Anda. Local Zones memiliki koneksi sendiri ke internet dan mendukung [AWS Direct Connect](#). Sumber daya yang dibuat di Local Zone dapat melayani pengguna lokal dengan komunikasi latensi sangat rendah. Untuk informasi selengkapnya, lihat [Panduan Pengguna AWS Local Zones](#) dan [Local Zones](#) di Panduan EC2 Pengguna Amazon.

Amazon EKS mendukung sumber daya tertentu di Local Zones. Ini termasuk [grup node terkelola](#), [EC2 node Amazon yang dikelola sendiri](#), volume Amazon EBS, dan Application Load Balancers (). ALBs Kami menyarankan Anda mempertimbangkan hal berikut saat menggunakan Local Zones sebagai bagian dari cluster Amazon EKS Anda.

- Anda tidak dapat membuat node Fargate di Local Zones dengan Amazon EKS.
- Pesawat kontrol Kubernetes yang dikelola Amazon EKS selalu berjalan di Wilayah. AWS Pesawat kontrol Kubernetes yang dikelola Amazon EKS tidak dapat berjalan di Zona Lokal. Karena Local Zones muncul sebagai subnet dalam VPC Anda, Kubernetes melihat sumber daya Local Zone Anda sebagai bagian dari subnet tersebut.
- [Cluster Amazon EKS Kubernetes berkomunikasi dengan EC2 instans Amazon yang Anda jalankan di AWS Wilayah atau Zona Lokal menggunakan antarmuka jaringan elastis terkelola Amazon EKS](#). Untuk mempelajari selengkapnya tentang arsitektur jaringan Amazon EKS, lihat [Mengkonfigurasi jaringan](#).
- Tidak seperti subnet regional, Amazon EKS tidak dapat menempatkan antarmuka jaringan ke subnet Zona Lokal Anda. Ini berarti bahwa Anda tidak harus menentukan subnet Local Zone ketika

Anda membuat klaster Anda. Namun, Anda dapat memiliki node pekerja di beberapa Local Zones yang berbeda yang terhubung ke cluster yang sama.

Memecahkan masalah dengan kluster dan node Amazon EKS

Bab ini mencakup beberapa kesalahan umum yang mungkin dapat Anda temui saat menggunakan Amazon EKS dan cara yang dilakukan untuk mengatasinya. Jika Anda perlu memecahkan masalah area Amazon EKS tertentu, lihat topik terpisah [the section called “Pemecahan Masalah”](#) [the section called “Memecahkan Masalah Konektor EKS”](#), dan [Pemecahan Masalah untuk ADOT menggunakan topik Eks Add-On](#).

Untuk informasi pemecahan masalah lainnya, lihat [konten Pusat Pengetahuan tentang Amazon Elastic Kubernetes Service di re:Post](#). AWS

Kapasitas tidak mencukupi

Jika Anda menerima kesalahan berikut saat mencoba membuat kluster Amazon EKS, maka salah satu Availability Zone yang Anda tentukan tidak memiliki kapasitas yang cukup untuk mendukung kluster.

```
Cannot create cluster 'example-cluster' because region-1d, the targeted Availability Zone, does not currently have sufficient capacity to support the cluster. Retry and choose from these Availability Zones: region-1a, region-1b, region-1c
```

Mencoba lagi membuat kluster Anda dengan subnet di kluster VPC yang meng-host di Availability Zone dikembalikan oleh pesan kesalahan ini.

Ada Availability Zone dimana cluster tidak dapat berada. Bandingkan Availability Zones tempat subnet Anda berada dengan daftar Availability Zone dalam [persyaratan dan pertimbangan Subnet](#).

Simpul gagal untuk bergabung dengan kluster

Ada beberapa alasan umum yang mencegah simpul bergabung dengan kluster:

- Jika node adalah node yang dikelola, Amazon EKS menambahkan entri ke `aws-auth ConfigMap` saat Anda membuat grup node. Jika entri dihapus atau dimodifikasi, maka Anda perlu menambahkannya kembali. Untuk informasi lebih lanjut, masukkan `eksctl create iamidentitymapping --help` di terminal Anda. Anda dapat melihat `aws-auth ConfigMap` entri Anda saat ini dengan mengganti `my-cluster` perintah berikut dengan nama cluster Anda

dan kemudian menjalankan perintah yang dimodifikasi: `eksctl get iamidentitymapping --cluster my-cluster`. ARN peran yang Anda tentukan tidak dapat menyertakan [jalur](#) selain `/`. Misalnya, jika nama peran Anda `development/apps/my-role`, Anda harus mengubahnya menjadi `my-role` saat menentukan ARN untuk peran tersebut. Pastikan Anda menentukan ARN peran IAM node (bukan ARN profil instance).

Jika node dikelola sendiri, dan Anda belum membuat [entri akses](#) untuk ARN peran IAM node, jalankan perintah yang sama yang terdaftar untuk node terkelola. Jika Anda telah membuat entri akses untuk ARN untuk peran IAM node Anda, maka itu mungkin tidak dikonfigurasi dengan benar di entri akses. Pastikan bahwa ARN peran IAM node (bukan ARN profil instance) ditentukan sebagai ARN utama dalam entri atau entri akses Anda. `aws-auth ConfigMap` Untuk informasi selengkapnya tentang entri akses, lihat [the section called “Berikan izin”](#).

- AWS CloudFormation Template `ClusterName` di node Anda tidak sama persis dengan nama cluster yang Anda inginkan untuk bergabung dengan node Anda. Melewati nilai yang salah ke bidang ini menghasilkan konfigurasi `/var/lib/kubelet/kubeconfig` file node yang salah, dan node tidak akan bergabung dengan cluster.
- Simpul tidak ditandai sebagai dimiliki oleh klaster. Simpul Anda harus memiliki tanda berikut yang diterapkan untuk mereka, di mana *my-cluster* diganti dengan nama klaster Anda.

Kunci	Nilai
<code>kubernetes.io/cluster/<i>my-cluster</i></code>	<code>owned</code>

- Simpul mungkin tidak dapat mengakses klaster menggunakan alamat IP publik. Pastikan bahwa simpul yang di-deploy di subnet publik memiliki alamat IP publik. Jika tidak, Anda dapat mengaitkan alamat IP Elastis ke node setelah diluncurkan. Untuk informasi selengkapnya, lihat [Mengaitkan alamat IP Elastis dengan instans berjalan atau antarmuka jaringan](#). Jika subnet publik tidak diatur secara otomatis untuk menetapkan alamat IP publik ke instans yang di-deploy untuk itu, maka kami merekomendasikan untuk mengaktifkan pengaturan tersebut. Untuk informasi selengkapnya, lihat [Memodifikasi atribut IPv4 pengalamatan publik untuk subnet Anda](#). Jika simpul di-deploy ke subnet privat, maka subnet harus memiliki rute ke gateway NAT yang memiliki alamat IP publik yang ditugaskan untuk itu.
- Titik akhir AWS STS untuk AWS Wilayah tempat Anda menerapkan node tidak diaktifkan untuk akun Anda. Untuk mengaktifkan wilayah, lihat [Mengaktifkan dan menonaktifkan AWS STS di suatu Wilayah AWS](#).

- Node tidak memiliki entri DNS pribadi, sehingga kubelet log berisi node `"" not found` kesalahan. Pastikan bahwa VPC tempat node dibuat memiliki nilai yang ditetapkan untuk domain-name dan domain-name-servers seperti Options dalam file. DHCP options set Nilai defaultnya adalah domain-name:<region>.compute.internal dan domain-name-servers:AmazonProvidedDNS. Untuk informasi selengkapnya, lihat [Set opsi DHCP](#) di Panduan Pengguna Amazon VPC.
- Jika node dalam grup node terkelola tidak terhubung ke cluster dalam waktu 15 menit, masalah kesehatan "NodeCreationFailure" akan dipancarkan dan status konsol akan disetel ke. Create failed Untuk Windows AMIs yang memiliki waktu peluncuran lambat, masalah ini dapat diselesaikan menggunakan [peluncuran cepat](#).

Untuk mengidentifikasi dan memecahkan masalah penyebab umum yang mencegah node pekerja bergabung dengan cluster, Anda dapat menggunakan runbook. `AWSSupport-TroubleshootEKSTaskWorkerNode` Untuk informasi selengkapnya, lihat [AWSSupport-TroubleshootEKSTaskWorkerNode](#) di referensi buku runbook Automation AWS Systems Manager.

Tidak sah atau akses ditolak (**kubectl**)

Jika Anda menerima salah satu kesalahan berikut saat menjalankan `kubectl` perintah, maka Anda belum `kubectl` mengonfigurasi dengan benar untuk Amazon EKS atau kredensi untuk prinsipal IAM (peran atau pengguna) yang Anda gunakan tidak dipetakan ke nama pengguna Kubernetes yang memiliki izin yang cukup untuk objek Kubernetes di kluster Amazon EKS Anda.

- `could not get token: AccessDenied: Access denied`
- `error: You must be logged in to the server (Unauthorized)`
- `error: the server doesn't have a resource type "svc"`

Ini bisa disebabkan oleh salah satu alasan berikut:

- Cluster dibuat dengan kredensi untuk satu prinsipal IAM dan `kubectl` dikonfigurasi untuk menggunakan kredensi untuk prinsipal IAM yang berbeda. Untuk mengatasi hal ini, perbarui `kube config` file Anda untuk menggunakan kredensi yang membuat kluster. Untuk informasi selengkapnya, lihat [the section called "Akses kluster dengan kubectl"](#).
- Jika kluster Anda memenuhi persyaratan platform minimum di bagian prasyarat [akses pengguna Grant IAM ke Kubernetes dengan entri akses EKS, entri akses tidak ada dengan](#) prinsipal IAM

Anda. Jika ada, itu tidak memiliki nama grup Kubernetes yang diperlukan yang ditentukan untuknya, atau tidak memiliki kebijakan akses yang tepat yang terkait dengannya. Untuk informasi selengkapnya, lihat [the section called “Berikan izin”](#).

- Jika klaster Anda tidak memenuhi persyaratan platform minimum di [Grant IAM pengguna akses ke Kubernetes dengan entri akses EKS, entri](#) dengan prinsipal IAM Anda tidak ada di `aws-auth ConfigMap` Jika ada, itu tidak dipetakan ke nama grup Kubernetes yang terikat ke Kubernetes atau dengan izin yang diperlukan `Role`. `ClusterRole` [Untuk informasi selengkapnya tentang objek Kubernetes role-based authorization \(RBAC\), lihat Menggunakan otorisasi RBAC dalam dokumentasi Kubernetes](#). Anda dapat melihat `aws-auth ConfigMap` entri Anda saat ini dengan mengganti `my-cluster` perintah berikut dengan nama cluster Anda dan kemudian menjalankan perintah yang dimodifikasi: `eksctl get iamidentitymapping --cluster my-cluster`. Jika entri untuk ARN kepala sekolah IAM Anda tidak ada di `ConfigMap`, masukkan `eksctl create iamidentitymapping --help` terminal Anda untuk mempelajari cara membuatnya.

Jika Anda menginstal dan mengkonfigurasi AWS CLI, Anda dapat mengonfigurasi kredensi IAM yang Anda gunakan. Untuk informasi selengkapnya, lihat [Mengonfigurasi AWS CLI](#) di Panduan Pengguna Antarmuka Baris AWS Perintah. Anda juga dapat mengonfigurasi `kubectl` untuk menggunakan peran IAM, jika Anda mengambil peran IAM untuk mengakses objek Kubernetes di klaster Anda. Untuk informasi selengkapnya, lihat [the section called “Akses klaster dengan kubectl”](#).

hostname doesn't match

Versi Python sistem Anda harus 2.7.9 atau lebih baru. Jika tidak, Anda menerima `hostname doesn't match` kesalahan dengan panggilan AWS CLI ke Amazon EKS. Untuk informasi selengkapnya, lihat [Apa itu kesalahan “nama host tidak cocok”? di Python Permintaan Pertanyaan yang Sering Diajukan](#).

getsockopt: no route to host

Docker berjalan di Kisaran CIDR `172.17.0.0/16` di klaster Amazon EKS. Kami menyarankan agar subnet VPC cluster Anda tidak tumpang tindih dengan rentang ini. Jika tidak, Anda akan menerima kesalahan berikut:

```
Error: : error upgrading connection: error dialing backend: dial tcp
172.17.<nn>.<nn>:10250: getsockopt: no route to host
```

Instances failed to join the Kubernetes cluster

Jika Anda menerima kesalahan `Instances failed to join the Kubernetes cluster` dalam AWS Management Console, pastikan bahwa akses endpoint pribadi kluster diaktifkan, atau bahwa Anda telah mengonfigurasi blok CIDR dengan benar untuk akses titik akhir publik. Untuk informasi selengkapnya, lihat [the section called “Akses titik akhir klaster”](#).

Kode kesalahan grup node terkelola

Jika grup node terkelola Anda mengalami masalah kesehatan perangkat keras, Amazon EKS mengembalikan kode kesalahan untuk membantu Anda mendiagnosis masalah tersebut. Pemeriksaan kesehatan ini tidak mendeteksi masalah perangkat lunak karena didasarkan pada [pemeriksaan EC2 kesehatan Amazon](#). Daftar berikut menjelaskan kode kesalahan.

AccessDenied

Amazon EKS atau satu atau beberapa node terkelola Anda gagal mengautentikasi atau mengotorisasi dengan server API cluster Kubernetes Anda. Untuk informasi lebih lanjut tentang menyelesaikan penyebab umum, lihat [the section called “Memperbaiki penyebab umum AccessDenied kesalahan untuk grup node terkelola”](#). Windows pribadi juga AMIs dapat menyebabkan kode kesalahan ini di samping pesan `Not authorized for images` kesalahan. Untuk informasi selengkapnya, lihat [the section called “Not authorized for images”](#).

AmiIdNotFound

Kami tidak dapat menemukan ID AMI yang terkait dengan template peluncuran Anda. Pastikan bahwa AMI ada dan dibagikan dengan akun Anda.

AutoScalingGroupNotFound

Kami tidak dapat menemukan grup Auto Scaling yang terkait dengan grup node terkelola. Anda dapat membuat ulang grup Auto Scaling dengan pengaturan yang sama untuk memulihkan.

ClusterUnreachable

Amazon EKS atau satu atau beberapa node terkelola Anda tidak dapat berkomunikasi dengan server API cluster Kubernetes Anda. Hal ini dapat terjadi jika ada gangguan jaringan atau jika waktu keluar permintaan pemrosesan server API.

Ec2 SecurityGroupNotFound

Kami tidak dapat menemukan grup keamanan cluster untuk cluster. Anda harus membuat ulang klaster Anda.

Ec2 SecurityGroupDeletionFailure

Kami tidak dapat menghapus grup keamanan akses jarak jauh untuk grup node terkelola Anda. Menghapus dependensi dari grup keamanan.

Ec2 LaunchTemplateNotFound

Kami tidak dapat menemukan template EC2 peluncuran Amazon untuk grup node terkelola Anda. Anda harus membuat ulang grup node Anda untuk memulihkan.

Ec2 LaunchTemplateVersionMismatch

Versi template EC2 peluncuran Amazon untuk grup node terkelola Anda tidak cocok dengan versi yang dibuat Amazon EKS. Anda mungkin dapat kembali ke versi yang dibuat Amazon EKS untuk memulihkan.

IamInstanceProfileNotFound

Kami tidak dapat menemukan profil instans IAM untuk grup node terkelola Anda. Anda mungkin dapat membuat ulang profil instans dengan pengaturan yang sama untuk memulihkan.

IamNodeRoleNotFound

Kami tidak dapat menemukan peran IAM untuk grup node terkelola Anda. Anda mungkin dapat membuat ulang IAM role dengan pengaturan yang sama untuk memulihkan.

AsgInstanceLaunchFailures

Grup Auto Scaling Anda mengalami kegagalan saat mencoba meluncurkan instance.

NodeCreationFailure

Instans yang diluncurkan tidak dapat mendaftar dengan kluster Amazon EKS Anda. Penyebab umum kegagalan ini adalah tidak mencukupi [IAM role simpul](#) izin atau kurangnya akses internet keluar untuk simpul. Node Anda harus memenuhi salah satu dari persyaratan berikut:

- Mampu mengakses internet menggunakan alamat IP publik. Kelompok keamanan yang terkait dengan subnet tempat node berada harus mengizinkan komunikasi. Untuk informasi selengkapnya, lihat [the section called “Persyaratan dan pertimbangan subnet”](#) dan [the section called “Persyaratan grup keamanan”](#).

- Node dan VPC Anda harus memenuhi persyaratan dalam [Menyebarkan kluster pribadi dengan akses internet terbatas](#).

InstanceLimitExceeded

AWS Akun Anda tidak dapat meluncurkan instance lagi dari jenis instans yang ditentukan. Anda mungkin dapat meminta peningkatan batas EC2 instans Amazon untuk dipulihkan.

InsufficientFreeAddresses

Satu atau beberapa subnet yang terkait dengan grup node terkelola Anda tidak memiliki cukup alamat IP yang tersedia untuk node baru.

InternalFailure

Kesalahan ini biasanya disebabkan oleh masalah sisi server Amazon EKS.

Memperbaiki penyebab umum **AccessDenied** kesalahan untuk grup node terkelola

Penyebab paling umum kesalahan AccessDenied saat melakukan pengoperasian pada grup simpul yang terkelola adalah hilangnya eks:node-manager ClusterRole atau ClusterRoleBinding. Amazon EKS mengatur sumber daya ini di klaster Anda sebagai bagian dari orientasi dengan grup simpul yang terkelola, dan ini diperlukan untuk mengelola grup simpul.

ClusterRole mungkin berubah seiring berjalannya waktu, tetapi seharusnya terlihat serupa dengan contoh berikut:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: eks:node-manager
rules:
- apiGroups:
  - ''
  resources:
  - pods
  verbs:
  - get
  - list
  - watch
  - delete
- apiGroups:
  - ''
```

```
resources:
- nodes
verbs:
- get
- list
- watch
- patch
- apiGroups:
- ''
resources:
- pods/eviction
verbs:
- create
```

ClusterRoleBinding mungkin berubah seiring berjalannya waktu, tetapi seharusnya terlihat serupa dengan contoh berikut:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: eks:node-manager
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: eks:node-manager
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: eks:node-manager
```

Verifikasi bahwa `eks:node-manager ClusterRole` ada.

```
kubectl describe clusterrole eks:node-manager
```

Jika ada, membandingkan output dengan contoh `ClusterRole` sebelumnya.

Verifikasi bahwa `eks:node-manager ClusterRoleBinding` ada.

```
kubectl describe clusterrolebinding eks:node-manager
```

Jika ada, membandingkan output dengan contoh `ClusterRoleBinding` sebelumnya.

Jika Anda telah mengidentifikasi yang hilang atau rusak ClusterRole atau ClusterRoleBinding sebagai penyebab AccessDenied kesalahan saat meminta operasi grup node terkelola, Anda dapat memulihkannya. Simpan isi berikut ke file bernama *eks-node-manager-role.yaml*.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: eks:node-manager
rules:
- apiGroups:
  - ''
  resources:
  - pods
  verbs:
  - get
  - list
  - watch
  - delete
- apiGroups:
  - ''
  resources:
  - nodes
  verbs:
  - get
  - list
  - watch
  - patch
- apiGroups:
  - ''
  resources:
  - pods/eviction
  verbs:
  - create
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: eks:node-manager
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: eks:node-manager
subjects:
```

```
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: eks:node-manager
```

Terapkan file.

```
kubectl apply -f eks-node-manager-role.yaml
```

Coba lagi mengoperasikan grup simpul untuk melihat apakah itu menyelesaikan masalah Anda.

Not authorized for images

Salah satu penyebab potensial pesan `Not authorized for images` kesalahan adalah menggunakan Amazon EKS Windows AMI pribadi untuk meluncurkan grup node terkelola Windows. Setelah merilis Windows baru AMIs, AWS membuat AMIs yang lebih tua dari 4 bulan pribadi, yang membuat mereka tidak lagi dapat diakses. Jika grup node terkelola Anda menggunakan AMI Windows pribadi, pertimbangkan untuk [memperbarui grup node terkelola Windows Anda](#). Meskipun kami tidak dapat menjamin bahwa kami dapat menyediakan akses ke AMIs yang telah dibuat pribadi, Anda dapat meminta akses dengan mengajukan tiket ke AWS Support. Untuk informasi selengkapnya, lihat [Patch](#) di Panduan EC2 Pengguna Amazon.

Node dalam **NotReady** keadaan

Jika node Anda memasuki `NotReady` status, ini mungkin menunjukkan bahwa node tidak sehat dan tidak tersedia untuk menjadwalkan Pod baru. Hal ini dapat terjadi karena berbagai alasan, seperti node kekurangan sumber daya yang cukup untuk CPU, memori, atau ruang disk yang tersedia.

Untuk Windows yang dioptimalkan Amazon EKS AMIs, tidak ada reservasi untuk sumber daya komputasi yang ditentukan secara default dalam kubelet konfigurasi. [Untuk membantu mencegah masalah sumber daya, Anda dapat mencadangkan sumber daya komputasi untuk proses sistem kubelet dengan menyediakan nilai konfigurasi untuk kube-reserved dan/atau system-reserved.](#) Anda melakukan ini menggunakan parameter `-KubeletExtraArgs` baris perintah dalam skrip bootstrap. Untuk informasi selengkapnya, lihat [Reserve Compute Resources for System Daemon](#) dalam dokumentasi Kubernetes dan di panduan pengguna ini. [the section called “Parameter konfigurasi skrip bootstrap”](#)

EKS Log Kolektor

Untuk memecahkan masalah dengan node Amazon EKS, ada skrip pra-bangun yang tersedia di node yang terletak di `/etc/eks/log-collector-script/eks-log-collector.sh`. Anda dapat menggunakan skrip untuk mengumpulkan log diagnostik untuk kasus dukungan dan pemecahan masalah umum.

Menggunakan perintah berikut ini untuk menjalankan skrip di simpul Anda:

```
sudo bash /etc/eks/log-collector-script/eks-log-collector.sh
```

Note

Jika naskah tidak ada di lokasi itu. Anda dapat mengunduh dan menjalankan skrip secara manual dengan perintah berikut:

```
curl -O https://amazon-eks.s3.amazonaws.com/support/log-collector-script/linux/eks-log-collector.sh
sudo bash eks-log-collector.sh
```

Skrip mengumpulkan informasi diagnostik berikut.

```
$ sudo bash /etc/eks/log-collector-script/eks-log-collector.sh
```

```
    This is version 0.7.8. New versions can be found at https://github.com/aws-labs/
amazon-eks-ami/blob/main/log-collector-script/
```

```
Trying to collect common operating system logs...
```

```
Trying to collect kernel logs...
```

```
Trying to collect mount points and volume information...
```

```
...
```

```
...
```

```
Done... your bundled logs are located in /var/log/eks_i-EXAMPLE_2025-03-25_0000-
UTC_0.7.8.tar.gz
```

Informasi diagnostik dikumpulkan dan disimpan di:

```
/var/log/eks_i-EXAMPLE_2025-03-25_0000-UTC_0.7.8.tar.gz
```

[Untuk mengambil bundel log untuk node Bottlerocket, silakan merujuk ke Bottlerocket Log untuk lebih jelasnya.](#)

Jaringan waktu aktif kontainer belum siap

Anda mungkin mendapatkan kesalahan `Container runtime network not ready` dan otorisasi kesalahan yang mirip dengan berikut ini:

```
4191 kubelet.go:2130] Container runtime network not ready: NetworkReady=false
      reason:NetworkPluginNotReady message:docker: network plugin is not ready: cni config
      uninitialized
4191 reflector.go:205] k8s.io/kubernetes/pkg/kubelet/kubelet.go:452: Failed to list
      *v1.Service: Unauthorized
4191 kubelet_node_status.go:106] Unable to register node
      "ip-10-40-175-122.ec2.internal" with API server: Unauthorized
4191 reflector.go:205] k8s.io/kubernetes/pkg/kubelet/kubelet.go:452: Failed to list
      *v1.Service: Unauthorized
```

Ini dapat terjadi karena salah satu alasan berikut:

1. Anda juga tidak memiliki `aws-auth ConfigMap` di cluster Anda atau tidak menyertakan entri untuk peran IAM yang Anda konfigurasikan dengan node Anda.

Untuk mengatasi masalah ini, lihat entri yang ada di `ConfigMap` dengan mengganti `my-cluster` perintah berikut dengan nama cluster Anda, lalu jalankan perintah yang dimodifikasi: `eksctl get iamidentitymapping --cluster my-cluster`. Jika Anda menerima pesan kesalahan dari perintah, itu mungkin karena klaster Anda tidak memiliki file `aws-authConfigMap`. Perintah berikut menambahkan entri ke `ConfigMap`. Jika `ConfigMap` tidak ada, perintah juga membuatnya. Ganti `111122223333` dengan ID AWS akun untuk peran IAM dan `myAmazonEKSNodeRole` dengan nama peran node Anda.

```
eksctl create iamidentitymapping --cluster my-cluster \
  --arn arn:aws:iam::111122223333:role/myAmazonEKSNodeRole --group
  system:bootstrappers,system:nodes \
  --username system:node:{{EC2PrivateDNSName}}
```

ARN peran yang Anda tentukan tidak dapat menyertakan [jalur](#) selain. / Misalnya, jika nama peran Anda `development/apps/my-role`, Anda harus mengubahnya menjadi `my-role` saat menentukan ARN peran tersebut. Pastikan Anda menentukan ARN peran IAM node (bukan ARN profil instance).

2. Node yang dikelola sendiri berada dalam kluster dengan versi platform pada versi minimum yang tercantum dalam prasyarat dalam [akses pengguna Grant IAM ke Kubernetes dengan topik entri akses EKS, tetapi entri](#) tidak tercantum dalam `aws-auth ConfigMap` (lihat item sebelumnya) untuk peran IAM node atau entri akses tidak ada untuk peran tersebut. Untuk mengatasi masalah ini, lihat entri akses yang ada dengan mengganti `my-cluster` perintah berikut dengan nama cluster Anda, lalu jalankan perintah yang dimodifikasi: `aws eks list-access-entries --cluster-name my-cluster`. Perintah berikut menambahkan entri akses untuk peran IAM node. Ganti `111122223333` dengan ID AWS akun untuk peran IAM dan `myAmazonEKSTNodeRole` dengan nama peran node Anda. Jika Anda memiliki node Windows, ganti `EC2_LINUX` dengan `EC2_Windows`. Pastikan Anda menentukan ARN peran IAM node (bukan ARN profil instance).

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn arn:aws:iam::111122223333:role/myAmazonEKSTNodeRole --type EC2_LINUX
```

Waktu habis handshake TLS

Ketika node tidak dapat membuat koneksi ke titik akhir server API publik, Anda mungkin melihat kesalahan yang mirip dengan kesalahan berikut.

```
server.go:233] failed to run Kubelet: could not init cloud provider "aws": error finding instance i-1111f222f333e44c: "error listing AWS instances: \"RequestError: send request failed\\ncaused by: Post net/http: TLS handshake timeout\""
```

Proses `kubelet` akan terus-menerus me-respawn dan menguji titik akhir server API. Kesalahan juga dapat terjadi sementara selama prosedur yang performa update kluster bergulir di bidang kendali, seperti perubahan konfigurasi atau versi update.

Untuk mengatasi masalah itu, memeriksa tabel rute dan grup keamanan untuk memastikan bahwa lalu lintas dari simpul dapat mencapai titik akhir publik.

InvalidClientTokenId

Jika Anda menggunakan peran IAM untuk akun layanan untuk Pod atau DaemonSet di-deploy ke kluster di AWS Wilayah Tiongkok, dan belum menyetel variabel `AWS_DEFAULT_REGION` lingkungan dalam spesifikasi, Pod tersebut atau DaemonSet mungkin menerima kesalahan berikut:

```
An error occurred (InvalidClientTokenId) when calling the GetCallerIdentity operation:  
The security token included in the request is invalid
```

Untuk mengatasi masalah ini, Anda perlu menambahkan variabel `AWS_DEFAULT_REGION` lingkungan ke Pod atau DaemonSet spesifikasi Anda, seperti yang ditunjukkan pada contoh spesifikasi Pod berikut.

```
apiVersion: v1  
kind: Pod  
metadata:  
  name: envar-demo  
  labels:  
    purpose: demonstrate-envvars  
spec:  
  containers:  
    - name: envar-demo-container  
      image: gcr.io/google-samples/node-hello:1.0  
      env:  
        - name: AWS_DEFAULT_REGION  
          value: "region-code"
```

Grup node harus cocok dengan versi Kubernetes sebelum memutakhirkan bidang kontrol

Sebelum Anda memutakhirkan control plane ke versi Kubernetes baru, versi minor dari node terkelola dan Fargate di kluster Anda harus sama dengan versi versi control plane Anda saat ini. Amazon EKS `update-cluster-version` API menolak permintaan hingga Anda memutakhirkan semua node yang dikelola Amazon EKS ke versi cluster saat ini. Amazon EKS menyediakan APIs untuk memutakhirkan node terkelola. Untuk informasi tentang memutakhirkan versi Kubernetes grup node terkelola, lihat [the section called “Perbarui”](#). Untuk memutakhirkan versi node Fargate, hapus pod yang diwakili oleh node dan deploy ulang pod setelah Anda memutakhirkan bidang kontrol Anda. Untuk informasi selengkapnya, lihat [the section called “Perbarui versi Kubernetes”](#).

Saat meluncurkan banyak node, ada **Too Many Requests** kesalahan

Jika Anda meluncurkan banyak node secara bersamaan, Anda mungkin melihat pesan kesalahan di log eksekusi [data EC2 pengguna Amazon](#) yang mengatakan Too Many Requests. Ini dapat terjadi karena bidang kontrol sedang kelebihan beban dengan describeCluster panggilan. Kelebihan beban mengakibatkan pelambatan, node gagal menjalankan skrip bootstrap, dan node gagal bergabung dengan cluster sama sekali.

Pastikan bahwa `--apiserver-endpoint`, `--b64-cluster-ca`, dan `--dns-cluster-ip` argumen sedang diteruskan ke skrip bootstrap node. Saat memasukkan argumen ini, skrip bootstrap tidak perlu melakukan describeCluster panggilan, yang membantu mencegah bidang kontrol kelebihan beban. Untuk informasi selengkapnya, lihat [the section called “Berikan data pengguna untuk meneruskan argumen ke bootstrap.sh file yang disertakan dengan AMI Linux/Bottlerocket Amazon EKS yang dioptimalkan”](#).

Respons kesalahan HTTP 401 tidak sah pada permintaan server API Kubernetes

Anda akan melihat kesalahan ini jika token akun layanan Pod telah kedaluwarsa di sebuah klaster.

Server API Kubernetes dari klaster Amazon EKS Anda menolak permintaan dengan token yang lebih lama dari 90 hari. Di versi Kubernetes sebelumnya, token tidak memiliki kedaluwarsa. Ini berarti bahwa klien yang mengandalkan token ini harus menyegarkannya dalam waktu satu jam. Untuk mencegah server Kubernetes API menolak permintaan Anda karena token tidak valid, versi [SDK klien Kubernetes](#) yang digunakan oleh beban kerja Anda harus sama, atau lebih lambat dari versi berikut:

- Versi Go 0.15.7 dan yang lebih baru
- Versi 12.0.0 Python dan yang lebih baru
- Versi Java 9.0.0 dan yang lebih baru
- JavaScript versi 0.10.3 dan yang lebih baru
- Cabang Ruby master
- Versi Haskell 0.3.0.0
- Versi C# 7.0.5 dan yang lebih baru

Anda dapat mengidentifikasi semua Pod yang ada di kluster Anda yang menggunakan token basi. Untuk informasi selengkapnya, lihat [the section called “Token akun layanan”](#).

Versi platform Amazon EKS lebih dari dua versi di belakang versi platform saat ini

Ini dapat terjadi ketika Amazon EKS tidak dapat memperbarui [versi platform](#) cluster Anda secara otomatis. Meskipun ada banyak penyebab untuk ini, beberapa penyebab umum mengikuti. Jika salah satu dari masalah ini berlaku untuk cluster Anda, mungkin masih berfungsi, versi platformnya tidak akan diperbarui oleh Amazon EKS.

Masalah

[Peran IAM cluster](#) telah dihapus — Peran ini ditentukan saat cluster dibuat. Anda dapat melihat peran mana yang ditentukan dengan perintah berikut. Ganti *my-cluster* dengan nama kluster Anda.

```
aws eks describe-cluster --name my-cluster --query cluster.roleArn --output text | cut -d / -f 2
```

Contoh output adalah sebagai berikut.

```
eksClusterRole
```

Solusi

Buat [peran IAM cluster](#) baru dengan nama yang sama.

Masalah

Subnet yang ditentukan selama pembuatan cluster telah dihapus — Subnet yang digunakan dengan cluster ditentukan selama pembuatan cluster. Anda dapat melihat subnet mana yang ditentukan dengan perintah berikut. Ganti *my-cluster* dengan nama kluster Anda.

```
aws eks describe-cluster --name my-cluster --query cluster.resourcesVpcConfig.subnetIds
```

Contoh output adalah sebagai berikut.

```
[  
  "subnet-EXAMPLE1",  
  "subnet-EXAMPLE2"
```

```
]
```

Solusi

Konfirmasikan apakah subnet IDs ada di akun Anda.

```
vpc_id=$(aws eks describe-cluster --name my-cluster --query
cluster.resourcesVpcConfig.vpcId --output text)
aws ec2 describe-subnets --filters "Name=vpc-id,Values=$vpc_id" --query
"Subnets[*].SubnetId"
```

Contoh output adalah sebagai berikut.

```
[
  "subnet-EXAMPLE3",
  "subnet-EXAMPLE4"
]
```

Jika subnet yang IDs dikembalikan dalam output tidak cocok dengan subnet IDs yang ditentukan saat cluster dibuat, maka jika Anda ingin Amazon EKS memperbarui cluster, Anda perlu mengubah subnet yang digunakan oleh cluster. Ini karena jika Anda menentukan lebih dari dua subnet saat membuat klaster, Amazon EKS secara acak memilih subnet yang Anda tentukan untuk membuat antarmuka jaringan elastis baru. Antarmuka jaringan ini memungkinkan bidang kontrol untuk berkomunikasi dengan node Anda. Amazon EKS tidak akan memperbarui cluster jika subnet yang dipilihnya tidak ada. Anda tidak memiliki kendali atas subnet mana yang Anda tentukan pada pembuatan cluster yang dipilih Amazon EKS untuk membuat antarmuka jaringan baru.

Saat Anda memulai pembaruan versi Kubernetes untuk klaster Anda, pembaruan dapat gagal karena alasan yang sama.

Masalah

Grup keamanan yang ditentukan selama pembuatan klaster telah dihapus - Jika Anda menentukan grup keamanan selama pembuatan klaster, Anda dapat melihatnya IDs dengan perintah berikut. Ganti *my-cluster* dengan nama klaster Anda.

```
aws eks describe-cluster --name my-cluster --query
cluster.resourcesVpcConfig.securityGroupIds
```

Contoh output adalah sebagai berikut.

```
[  
    "sg-EXAMPLE1"  
]
```

Jika [] dikembalikan, maka tidak ada grup keamanan yang ditentukan saat cluster dibuat dan grup keamanan yang hilang bukanlah masalahnya. Jika grup keamanan dikembalikan, maka konfirmasi bahwa grup keamanan ada di akun Anda.

Solusi

Konfirmasikan apakah grup keamanan ini ada di akun Anda.

```
vpc_id=$(aws eks describe-cluster --name my-cluster --query  
cluster.resourcesVpcConfig.vpcId --output text)  
aws ec2 describe-security-groups --filters "Name=vpc-id,Values=$vpc_id" --query  
"SecurityGroups[*].GroupId"
```

Contoh output adalah sebagai berikut.

```
[  
    "sg-EXAMPLE2"  
]
```

Jika grup keamanan yang IDs ditampilkan dalam output tidak cocok dengan grup keamanan IDs yang ditentukan saat kluster dibuat, maka jika Anda ingin Amazon EKS memperbarui cluster, Anda perlu mengubah grup keamanan yang digunakan oleh cluster. Amazon EKS tidak akan memperbarui kluster jika grup keamanan IDs yang ditentukan pada pembuatan kluster tidak ada.

Saat Anda memulai pembaruan versi Kubernetes untuk kluster Anda, pembaruan dapat gagal karena alasan yang sama.

- Anda tidak memiliki setidaknya enam (meskipun kami merekomendasikan 16) alamat IP yang tersedia di setiap subnet yang Anda tentukan saat Anda membuat cluster Anda. Jika Anda tidak memiliki cukup alamat IP yang tersedia di subnet, Anda perlu membebaskan alamat IP di subnet atau Anda perlu mengubah subnet yang digunakan oleh cluster untuk menggunakan subnet dengan alamat IP yang cukup tersedia.
- Anda mengaktifkan [enkripsi rahasia](#) saat membuat kluster dan kunci AWS KMS yang Anda tentukan telah dihapus. Jika Anda ingin Amazon EKS memperbarui cluster, Anda perlu membuat cluster baru

Kesehatan cluster FAQs dan kode kesalahan dengan jalur resolusi

Amazon EKS mendeteksi masalah dengan kluster EKS Anda dan infrastruktur kluster dan menyimpannya di objek kesehatan sumber daya kluster EKS Anda. Anda dapat mendeteksi, memecahkan masalah, dan mengatasi masalah kluster lebih cepat dengan bantuan informasi kesehatan kluster. Ini memungkinkan Anda untuk membuat lingkungan aplikasi yang lebih aman dan up-to-date. Selain itu, mungkin tidak mungkin bagi Anda untuk meningkatkan ke versi Kubernetes yang lebih baru atau Amazon EKS untuk menginstal pembaruan keamanan pada kluster yang terdegradasi sebagai akibat dari masalah dengan infrastruktur atau konfigurasi kluster yang diperlukan. Amazon EKS dapat memakan waktu 3 jam untuk mendeteksi masalah atau mendeteksi bahwa masalah telah teratasi.

Kesehatan kluster Amazon EKS adalah tanggung jawab bersama antara Amazon EKS dan penggunanya. Anda bertanggung jawab atas infrastruktur prasyarat peran IAM dan subnet Amazon VPC, serta infrastruktur lain yang diperlukan, yang harus disediakan terlebih dahulu. Amazon EKS mendeteksi perubahan dalam konfigurasi infrastruktur ini dan cluster.

Untuk mengakses kesehatan kluster Anda di konsol Amazon EKS, cari tabel bernama Masalah Kesehatan di tab Masalah kesehatan kluster di dasbor observabilitas yang diakses dari halaman detail kluster Amazon EKS. Data ini juga akan tersedia dengan memanggil `DescribeCluster` tindakan di EKS API, misalnya dari dalam Antarmuka Baris AWS Perintah.

Mengapa saya harus menggunakan fitur ini?

Anda akan mendapatkan peningkatan visibilitas ke dalam kesehatan kluster Amazon EKS Anda, dengan cepat mendiagnosis dan memperbaiki masalah apa pun, tanpa perlu menghabiskan waktu debugging atau membuka AWS kasus dukungan. Misalnya: Anda secara tidak sengaja menghapus subnet untuk kluster Amazon EKS, Amazon EKS tidak akan dapat membuat antarmuka jaringan lintas akun dan perintah AWS CLI Kubernetes seperti `exec` atau `log`. `kubectl kubectl` Ini akan gagal dengan kesalahan: `Error from server: error dialing backend: remote error: tls: internal error`. Sekarang Anda akan melihat masalah kesehatan Amazon EKS yang mengatakan: `subnet-da60e280 was deleted: could not create network interface`.

Bagaimana fitur ini berhubungan atau bekerja dengan AWS layanan lain?

Peran IAM dan subnet VPC Amazon adalah dua contoh infrastruktur prasyarat yang mendeteksi masalah kesehatan kluster. Fitur ini akan mengembalikan informasi terperinci jika sumber daya tersebut tidak dikonfigurasi dengan benar.

Apakah cluster dengan masalah kesehatan menimbulkan biaya?

Ya, setiap cluster Amazon EKS ditagih dengan harga Amazon EKS standar. Fitur kesehatan cluster tersedia tanpa biaya tambahan.

Apakah fitur ini berfungsi dengan cluster Amazon EKS di AWS Outposts?

Ya, masalah cluster terdeteksi untuk kluster EKS di AWS Cloud termasuk cluster yang diperluas di Outposts dan cluster lokal di AWS Outposts. AWS Kesehatan cluster tidak mendeteksi masalah dengan Amazon EKS Anywhere atau Amazon EKS Distro (EKS-D).

Bisakah saya mendapatkan pemberitahuan ketika masalah baru terdeteksi?

Ya. AWS mengirimkan email dan pemberitahuan Personal Health Dashboard ketika masalah kesehatan klaster baru terdeteksi.

Apakah konsol memberi saya peringatan untuk masalah kesehatan?

Ya, setiap cluster dengan masalah kesehatan akan menyertakan spanduk di bagian atas konsol.

Dua kolom pertama adalah apa yang dibutuhkan untuk nilai respons API. Bidang ketiga dari ClusterIssue objek [Health](#) adalah ResourceIds, yang pengembaliannya tergantung pada jenis masalah.

Kode	Pesan	ResourceIds	Cluster Dapat Dipulihkan?
SUBNET_NOT_FOUND	Kami tidak dapat menemukan satu atau lebih subnet yang saat ini terkait dengan cluster Anda. Hubungi Amazon EKS update-cluster-config API untuk memperbarui subnet.	Id Subnet	Ya
SECURITY_GROUP_NOT_FOUND	Kami tidak dapat menemukan satu atau lebih grup keamanan yang saat ini terkait	Id grup keamanan	Ya

Kode	Pesan	ResourceIds	Cluster Dapat Dipulihkan?
	dengan klaster Anda. Hubungi Amazon EKS update-cluster-config API untuk memperbaiki grup keamanan		
IP_NOT_TERSEDIA	Satu atau beberapa subnet yang terkait dengan cluster Anda tidak memiliki cukup alamat IP yang tersedia untuk Amazon EKS untuk melakukan operasi manajemen klaster. Kosongkan alamat di subnet, atau kaitkan subnet yang berbeda ke klaster Anda menggunakan Amazon EKS update-cluster-config API.	Id Subnet	Ya
VPC_NOT_FOUND	Kami tidak dapat menemukan VPC yang terkait dengan cluster Anda. Anda harus menghapus dan membuat ulang cluster Anda.	VPC id	Tidak

Kode	Pesan	ResourceIds	Cluster Dapat Dipulihkan?
ASSUME_ROLE_ACCESS_DENIED	Cluster Anda tidak menggunakan Amazon EKS service-linked-role. Kami tidak dapat mengambil peran yang terkait dengan klaster Anda untuk melakukan operasi manajemen Amazon EKS yang diperlukan. Periksa peran yang ada dan memiliki kebijakan kepercayaan yang diperlukan.	Peran IAM cluster	Ya

Kode	Pesan	ResourceIds	Cluster Dapat Dipulihkan?
PERMISSION_ACCESS_DENIED	Cluster Anda tidak menggunakan Amazon EKS service-linked-role. Peran yang terkait dengan klaster Anda tidak memberikan izin yang memadai bagi Amazon EKS untuk melakukan operasi manajemen yang diperlukan. Periksa kebijakan yang dilampirkan pada peran klaster dan jika ada kebijakan penolakan terpisah yang diterapkan.	Peran IAM cluster	Ya
ASSUME_ROLE_ACCESS_DENIED_USING_SLR	Kami tidak dapat mengasumsikan manajemen cluster Amazon EKS service-linked-role. Periksa peran yang ada dan memiliki kebijakan kepercayaan yang diperlukan.	Amazon EKS service-linked-role	Ya

Kode	Pesan	ResourceIds	Cluster Dapat Dipulihkan?
IZIN_ACCESS_DENIED_USING_SLR	Manajemen klaster Amazon EKS service-linked-role tidak memberikan izin yang cukup bagi Amazon EKS untuk melakukan operasi manajemen yang diperlukan. Periksa kebijakan yang dilampirkan pada peran klaster dan jika ada kebijakan penolakan terpisah yang diterapkan.	Amazon EKS service-linked-role	Ya
OPT_IN_REQUIRED	Akun Anda tidak memiliki langganan EC2 layanan Amazon. Perbarui langganan akun Anda di halaman pengaturan akun Anda.	N/A	Ya
STS_REGIONAL_ENDPOINT_DISABLED	Titik akhir regional STS dinonaktifkan. Aktifkan titik akhir Amazon EKS untuk melakukan operasi manajemen klaster yang diperlukan.	N/A	Ya

Kode	Pesan	ResourceIds	Cluster Dapat Dipulihkan?
KMS_KEY_DISABLED	Kunci AWS KMS yang terkait dengan cluster Anda dinonaktifkan. Aktifkan kembali kunci untuk memulihkan cluster Anda.	Kunci KMS Arn	Ya
KMS_KEY_NOT_FOUND	Kami tidak dapat menemukan kunci AWS KMS yang terkait dengan cluster Anda. Anda harus menghapus dan membuat ulang cluster.	ARN Kunci KMS	Tidak
KMS_GRANT_DICABUT	Hibah untuk Kunci AWS KMS yang terkait dengan cluster Anda dicabut. Anda harus menghapus dan membuat ulang cluster.	Kunci KMS Arn	Tidak

Hubungkan klaster Kubernetes ke Konsol Manajemen Amazon EKS dengan Konektor Amazon EKS

Anda dapat menggunakan Amazon EKS Connector untuk mendaftarkan dan menghubungkan klaster Kubernetes yang sesuai dan AWS memvisualisasikannya di konsol Amazon EKS. Setelah cluster terhubung, Anda dapat melihat status, konfigurasi, dan beban kerja untuk klaster tersebut di konsol Amazon EKS. Anda dapat menggunakan fitur ini untuk melihat kluster yang terhubung di konsol Amazon EKS, tetapi Anda tidak dapat mengelolanya. Konektor Amazon EKS membutuhkan agen yang merupakan [proyek open source di Github](#). Untuk konten teknis tambahan, termasuk pertanyaan umum dan pemecahan masalah, lihat [the section called “Memecahkan Masalah Konektor EKS”](#)

Konektor Amazon EKS dapat menghubungkan jenis cluster Kubernetes berikut ke Amazon EKS.

- Cluster Kubernetes lokal
- Cluster yang dikelola sendiri yang berjalan di Amazon EC2
- Cluster terkelola dari penyedia cloud lainnya

Pertimbangan Amazon EKS Connector

Sebelum Anda menggunakan Amazon EKS Connector, pahami hal berikut:

- Anda harus memiliki hak administratif ke klaster Kubernetes untuk menghubungkan cluster ke Amazon EKS.
- Cluster Kubernetes harus memiliki node pekerja Linux 64-bit (x86) sebelum menghubungkan. Node pekerja ARM tidak didukung.
- Anda harus memiliki node pekerja di klaster Kubernetes Anda yang memiliki akses keluar ke endpoint dan Systems ssm. Managersssmmessages.. Untuk informasi selengkapnya, lihat [titik akhir Systems Manager](#) di Referensi AWS Umum.
- Secara default, Anda dapat menghubungkan hingga 10 cluster di Region. Anda dapat meminta peningkatan melalui [konsol kuota layanan](#). Lihat [Meminta peningkatan kuota](#) untuk informasi selengkapnya.
- Hanya Amazon EKSRegisterCluster,, ListClustersDescribeCluster, dan yang DeregisterCluster APIs didukung untuk klaster Kubernetes eksternal.
- Anda harus memiliki izin berikut untuk mendaftarkan klaster:

- eks: RegisterCluster
 - ssm: CreateActivation
 - ssm: DeleteActivation
 - saya: PassRole
- Anda harus memiliki izin berikut untuk membatalkan pendaftaran kluster:
- eks: DeregisterCluster
 - ssm: DeleteActivation
 - ssm: DeregisterManagedInstance

Peran IAM yang diperlukan untuk Konektor Amazon EKS

Menggunakan Konektor Amazon EKS memerlukan dua peran IAM berikut:

- Peran terkait layanan [Amazon EKS Connector](#) dibuat saat Anda mendaftarkan kluster untuk pertama kalinya.
- Anda harus membuat peran IAM agen Amazon EKS Connector. Lihat [the section called “Peran konektor IAM”](#) untuk detail.

Untuk mengaktifkan izin tampilan kluster dan beban kerja untuk [prinsipal IAM, terapkan peran kluster Amazon EKS Connector](#) dan eks-connector Amazon ke kluster Anda. Ikuti langkah-langkah dalam [akses Grant untuk melihat resource cluster Kubernetes di konsol Amazon EKS](#).

Connect cluster Kubernetes eksternal ke Amazon EKS Management Console

Anda dapat menghubungkan cluster Kubernetes eksternal ke Amazon EKS dengan menggunakan beberapa metode dalam proses berikut. Proses ini melibatkan dua langkah: Mendaftarkan cluster dengan Amazon EKS dan menginstal eks-connector agen di cluster.

Important

Anda harus menyelesaikan langkah kedua dalam waktu 3 hari setelah menyelesaikan langkah pertama, sebelum pendaftaran berakhir.

Pertimbangan

Anda dapat menggunakan manifes YAMAL saat menginstal agen. Atau, Anda dapat menggunakan Helm jika Anda mendaftarkan cluster dengan AWS Management Console atau Antarmuka Baris AWS Perintah. Namun, Anda tidak dapat menggunakan Helm untuk menginstal agen jika Anda mendaftarkan cluster. `eksctl`

Prasyarat

- Pastikan peran agen Amazon EKS Connector telah dibuat. Ikuti langkah-langkah dalam [Membuat peran agen konektor Amazon EKS](#).
- Anda harus memiliki izin berikut untuk mendaftarkan klaster:
 - `eks:RegisterCluster`
 - `ssm:CreateActivation`
 - `ssm>DeleteActivation`
 - `iam:PassRole`

Langkah 1: Mendaftarkan cluster

Untuk mendaftarkan cluster ke konektor Amazon EKS, Anda dapat menggunakan salah satu alat berikut:

- [the section called “AWS CLI”](#)
- [the section called “AWS Management Console”](#)
- [the section called “eksctl”](#)

AWS CLI

1. AWS CLI harus diinstal. Untuk menginstal atau memutakhirkannya, lihat [Menginstal AWS CLI](#).
2. Untuk konfigurasi Konektor, tentukan peran IAM agen Amazon EKS Connector Anda. Untuk informasi selengkapnya, lihat [the section called “Peran IAM yang diperlukan untuk Konektor Amazon EKS”](#).

```
aws eks register-cluster \  
    --name my-first-registered-cluster \  
    --role arn:aws:iam::123456789012:role/eksctl-role
```

```
--connector-config roleArn=arn:aws: iam::111122223333:role/
AmazonEKSCoordinatorAgentRole,provider="OTHER" \
--region aws-region
```

Contoh output adalah sebagai berikut.

```
{
  "cluster": {
    "name": "my-first-registered-cluster",
    "arn": "arn:aws: eks:region:111122223333:cluster/my-first-registered-
cluster",
    "createdAt": 1627669203.531,
    "ConnectorConfig": {
      "activationId": "xxxxxxxxACTIVATION_IDxxxxxxxx",
      "activationCode": "xxxxxxxxACTIVATION_CODExxxxxxxx",
      "activationExpiry": 1627672543.0,
      "provider": "OTHER",
      "roleArn": "arn:aws: iam::111122223333:role/AmazonEKSCoordinatorAgentRole"
    },
    "status": "CREATING"
  }
}
```

Anda menggunakan `aws-region`, `activationId`, dan `activationCode` nilai di langkah berikutnya.

AWS Management Console

1. Buka [konsol Amazon EKS](#).
2. Pilih Add cluster dan pilih Register untuk membuka halaman konfigurasi.
3. Pada bagian Configure cluster, isi kolom berikut:
 - Nama — Nama unik untuk klaster Anda.
 - Provider — Pilih untuk menampilkan daftar dropdown dari penyedia klaster Kubernetes. Jika Anda tidak tahu penyedia tertentu, pilih Lainnya.
 - Peran Konektor EKS - Pilih peran yang akan digunakan untuk menghubungkan cluster.
4. Pilih Daftar cluster.

5. Halaman ikhtisar Cluster ditampilkan. Jika Anda ingin menggunakan bagan Helm, salin `helm install` perintah dan lanjutkan ke langkah berikutnya. Jika Anda ingin menggunakan manifes YAMM, pilih Unduh file YAMM untuk mengunduh file manifes ke drive lokal Anda.

Important

Ini adalah satu-satunya kesempatan Anda untuk menyalin `helm install` perintah atau mengunduh file ini. Jangan menavigasi jauh dari halaman ini, karena tautan tidak akan dapat diakses dan Anda harus membatalkan pendaftaran cluster dan memulai langkah-langkahnya dari awal.

Perintah atau file manifes hanya dapat digunakan sekali untuk cluster terdaftar. Jika Anda menghapus sumber daya dari klaster Kubernetes, Anda harus mendaftarkan ulang klaster dan mendapatkan file manifes baru.

Lanjutkan ke langkah berikutnya untuk menerapkan file manifes ke cluster Kubernetes Anda.

eksctl

1. `eksctl` versi 0.68 atau yang lebih baru harus diinstal. Untuk menginstal atau memutakhirkannya, lihat [the section called “Buat cluster \(eksctl\)”](#).
2. Daftarkan cluster dengan memberikan nama, penyedia, dan wilayah.

```
eksctl register cluster --name my-cluster --provider my-provider --region region-code
```

Contoh output:

```
2021-08-19 13:47:26 [#] creating IAM role "eksctl-20210819194112186040"
2021-08-19 13:47:26 [#] registered cluster "<name>" successfully
2021-08-19 13:47:26 [#] wrote file eks-connector.yaml to <current directory>
2021-08-19 13:47:26 [#] wrote file eks-connector-clusterrole.yaml to <current
  directory>
2021-08-19 13:47:26 [#] wrote file eks-connector-console-dashboard-full-access-
  group.yaml to <current directory>
2021-08-19 13:47:26 [!] note: "eks-connector-clusterrole.yaml" and "eks-connector-
  console-dashboard-full-access-group.yaml" give full EKS Console access to IAM
  identity "<aws-arn>", edit if required; read https://eksctl.io/usage/eks-connector
  for more info
```

```
2021-08-19 13:47:26 [#] run `kubectl apply -f eks-connector.yaml,eks-connector-clusterrole.yaml,eks-connector-console-dashboard-full-access-group.yaml` before expiry> to connect the cluster
```

Ini membuat file di komputer lokal Anda. File-file ini harus diterapkan ke cluster eksternal dalam waktu 3 hari, atau pendaftaran berakhir.

3. Di terminal yang dapat mengakses cluster, terapkan `eks-connector-binding.yaml` file:

```
kubectl apply -f eks-connector-binding.yaml
```

Langkah 2: Memasang **eks-connector** agen

Untuk menginstal `eks-connector` agen, gunakan salah satu alat berikut:

- [the section called “Helm”](#)
- [the section called “yaml”](#)

Helm

Note

Jika Anda mendaftarkan kluster `eksctl`, gunakan metode manifes YAMAL alih-alih metode bagan Helm.

1. Jika Anda menggunakan AWS CLI pada langkah sebelumnya, ganti `ACTIVATION_CODE` dan `ACTIVATION_ID` dalam perintah berikut dengan `activationId`, dan `activationCode` nilai masing-masing. Ganti `aws-region` dengan AWS Region yang Anda gunakan pada langkah sebelumnya. Kemudian jalankan perintah untuk menginstal `eks-connector` agen pada cluster pendaftaran:

```
$ helm install eks-connector \
  --namespace eks-connector \
  oci://public.ecr.aws/eks-connector/eks-connector-chart \
  --set eks.activationCode=ACTIVATION_CODE \
  --set eks.activationId=ACTIVATION_ID \
  --set eks.agentRegion=aws-region
```

Jika Anda menggunakan langkah sebelumnya, gunakan perintah yang Anda salin dari langkah sebelumnya yang memiliki nilai-nilai ini terisi. AWS Management Console

2. Periksa kesehatan eks -connector penerapan yang diinstal dan tunggu status cluster terdaftar di Amazon EKS. ACTIVE

yaml

Selesaikan koneksi dengan menerapkan file manifes Amazon EKS Connector ke cluster Kubernetes Anda. Untuk melakukan ini, Anda harus menggunakan metode yang dijelaskan sebelumnya. Jika manifes tidak diterapkan dalam tiga hari, pendaftaran Amazon EKS Connector akan kedaluwarsa. Jika koneksi cluster kedaluwarsa, cluster harus dideregistrasi sebelum menghubungkan cluster lagi.

1. Unduh file YAMM Konektor Amazon EKS.

```
curl -O https://amazon-eks.s3.us-west-2.amazonaws.com/eks-connector/manifests/eks-connector/latest/eks-connector.yaml
```

2. Edit file YAMM Konektor Amazon EKS untuk mengganti semua referensi %AWS_REGION% %EKS_ACTIVATION_ID%,aws-region, %EKS_ACTIVATION_CODE% dengan activationId,, dan activationCode dari output langkah sebelumnya.

Contoh perintah berikut dapat menggantikan nilai-nilai ini.

```
sed -i "s~%AWS_REGION%~$aws-region~g; s~%EKS_ACTIVATION_ID%~$EKS_ACTIVATION_ID~g; s~%EKS_ACTIVATION_CODE%~$(echo -n $EKS_ACTIVATION_CODE | base64)~g" eks-connector.yaml
```

Important

Pastikan kode aktivasi Anda dalam format base64.

3. Di terminal yang dapat mengakses cluster, Anda dapat menerapkan file manifes yang diperbarui dengan menjalankan perintah berikut:

```
kubectl apply -f eks-connector.yaml
```

4. Setelah manifes Amazon EKS Connector dan file YAMM yang mengikat peran diterapkan ke klaster Kubernetes Anda, konfirmasi bahwa klaster tersebut sekarang terhubung.

```
aws eks describe-cluster \
  --name "my-first-registered-cluster" \
  --region AWS_REGION
```

Outputnya harus mencakup `status=ACTIVE`.

5. (Opsional) Tambahkan tag ke cluster Anda. Untuk informasi selengkapnya, lihat [the section called “Pemberian tag pada sumber daya Anda”](#).

Langkah selanjutnya

Jika Anda memiliki masalah dengan langkah-langkah ini, lihat [the section called “Memecahkan Masalah Konektor EKS”](#).

Untuk memberikan akses [prinsipal IAM tambahan ke konsol Amazon EKS](#) untuk melihat sumber daya Kubernetes di kluster yang terhubung, lihat [the section called “Berikan akses ke cluster”](#)

Berikan akses untuk melihat resource kluster Kubernetes di konsol Amazon EKS

Berikan akses kepada [prinsipal IAM ke konsol Amazon EKS](#) untuk melihat informasi tentang resource Kubernetes yang berjalan di kluster Anda yang terhubung.

Prasyarat

[Prinsipal IAM](#) yang Anda gunakan untuk mengakses AWS Management Console harus memenuhi persyaratan berikut:

- Itu harus memiliki izin `eks:AccessKubernetesApi` IAM.
- Akun layanan Amazon EKS Connector dapat meniru prinsip IAM di cluster. Hal ini memungkinkan Amazon EKS Connector untuk memetakan prinsipal IAM ke pengguna Kubernetes.

Untuk membuat dan menerapkan peran kluster Amazon EKS Connector

1. Unduh templat peran `eks-connector` cluster.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/eks-connector/manifests/eks-connector-console-roles/eks-connector-clusterrole.yaml
```

2. Edit file YAMAL template peran cluster. Ganti referensi %IAM_ARN% dengan Nama Sumber Daya Amazon (ARN) kepala sekolah IAM Anda.
3. Terapkan peran klaster Amazon EKS Connector YAMAL ke klaster Kubernetes Anda.

```
kubectl apply -f eks-connector-clusterrole.yaml
```

Agar prinsipal IAM dapat melihat sumber daya Kubernetes di konsol Amazon EKS, prinsipal harus dikaitkan dengan Kubernetes `role` atau `clusterrole` dengan izin yang diperlukan untuk membaca sumber daya. Untuk informasi selengkapnya, lihat [Menggunakan Otorisasi RBAC](#) dalam dokumentasi Kubernetes.

Untuk mengkonfigurasi prinsipal IAM untuk mengakses cluster yang terhubung

1. Anda dapat mengunduh salah satu dari contoh file manifes ini untuk membuat `clusterrole` dan `clusterrolebinding` atau `a role` dan `rolebinding`, masing-masing:

Lihat sumber daya Kubernetes di semua ruang nama

- Peran `eks-connector-console-dashboard-full-access-clusterrole` cluster memberikan akses ke semua ruang nama dan sumber daya yang dapat divisualisasikan di konsol. Anda dapat mengubah `namarole`, `clusterrole` dan pengikatan yang sesuai sebelum menerapkannya ke cluster Anda. Gunakan perintah berikut untuk mengunduh file sampel.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/eks-connector/manifests/eks-connector-console-roles/eks-connector-console-dashboard-full-access-group.yaml
```

Melihat sumber daya Kubernetes di namespace tertentu

- Namespace dalam file ini adalah `default`, jadi jika Anda ingin menentukan namespace yang berbeda, edit file sebelum menerapkannya ke cluster Anda. Gunakan perintah berikut untuk mengunduh file sampel.


```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/eks-connector/manifests/eks-connector-console-roles/eks-connector-console-dashboard-restricted-access-group.yaml
```

2. Edit file YAMAL akses penuh atau akses terbatas untuk mengganti referensi %IAM_ARN% dengan Amazon Resource Name (ARN) kepala sekolah IAM Anda.
3. Terapkan akses penuh atau akses terbatas file YAMAL ke kluster Kubernetes Anda. Ganti nilai file YAMAL dengan milik Anda sendiri.

```
kubectl apply -f eks-connector-console-dashboard-full-access-group.yaml
```

Untuk melihat sumber daya Kubernetes di kluster Anda yang terhubung, lihat. [the section called “Akses sumber daya kluster”](#) Data untuk beberapa jenis sumber daya pada tab Sumber Daya tidak tersedia untuk kluster yang terhubung.

Membatalkan pendaftaran cluster Kubernetes dari konsol Amazon EKS

Jika Anda selesai menggunakan cluster yang terhubung, Anda dapat membatalkan pendaftarannya. Setelah dideregistrasi, cluster tidak lagi terlihat di konsol Amazon EKS.

Anda harus memiliki izin berikut untuk memanggil API DeregisterCluster:

- eks:DeregisterCluster
- ssm:DeleteActivation
- ssm:DeregisterManagedInstance

Proses ini melibatkan dua langkah: Membatalkan pendaftaran cluster dengan Amazon EKS dan menghapus instalasi agen eks-connector di cluster.

Deregister cluster Kubernetes

Untuk membatalkan pendaftaran cluster dari konektor Amazon EKS, Anda dapat menggunakan salah satu alat berikut:

- [the section called “AWS CLI”](#)

- [the section called “AWS Management Console”](#)
- [the section called “eksctl”](#)

AWS CLI

1. AWS CLI harus diinstal. Untuk menginstal atau memutakhirkannya, lihat [Menginstal AWS CLI](#).
2. Pastikan peran agen Amazon EKS Connector telah dibuat.
3. Deregister cluster yang terhubung.

```
aws eks deregister-cluster \  
  --name my-cluster \  
  --region region-code
```

AWS Management Console

1. Buka [konsol Amazon EKS](#).
2. Pilih Klaster.
3. Pada halaman Clusters, pilih cluster yang terhubung dan pilih Deregister.
4. Konfirmasikan bahwa Anda ingin membatalkan pendaftaran cluster.

eksctl

1. Instal eksctl versi 0.68 atau yang lebih baru. Untuk menginstal atau memutakhirkannya, lihat [the section called “Buat cluster \(eksctl\)”](#).
2. Pastikan peran agen Amazon EKS Connector telah dibuat.
3. Deregister cluster yang terhubung:

```
eksctl deregister cluster --name my-cluster
```

Bersihkan sumber daya di klaster Kubernetes

Untuk menghapus instalasi eks-connector agen, gunakan salah satu alat berikut:

- [the section called “helm”](#)

- [the section called “yaml”](#)

helm

Jalankan perintah berikut untuk menghapus instalasi agen.

```
helm -n eks-connector uninstall eks-connector
```

yaml

1. Hapus file YAMM Amazon EKS Connector dari kluster Kubernetes Anda.

```
kubectl delete -f eks-connector.yaml
```

2. Jika Anda membuat `clusterrole` atau `clusterrolebindings` untuk [prinsipal IAM](#) tambahan untuk mengakses kluster, hapus dari kluster Kubernetes Anda.

Memecahkan masalah Konektor Amazon EKS

Topik ini mencakup beberapa kesalahan umum yang mungkin Anda temui saat menggunakan Konektor Amazon EKS, termasuk petunjuk tentang cara mengatasinya dan solusinya.

Pemecahan masalah dasar

Bagian ini menjelaskan langkah-langkah untuk mendiagnosis masalah Amazon EKS Connector.

Periksa status Konektor Amazon EKS

Untuk memeriksa status Konektor Amazon EKS, ketik:

```
kubectl get pods -n eks-connector
```

Periksa log Konektor Amazon EKS

Amazon EKS Connector Pod terdiri dari tiga kontainer. Untuk mengambil log lengkap untuk semua kontainer ini sehingga Anda dapat memeriksanya, jalankan perintah berikut:

- `connector-init`

```
kubectl logs eks-connector-0 --container connector-init -n eks-connector
kubectl logs eks-connector-1 --container connector-init -n eks-connector
```

- **connector-proxy**

```
kubectl logs eks-connector-0 --container connector-proxy -n eks-connector
kubectl logs eks-connector-1 --container connector-proxy -n eks-connector
```

- **connector-agent**

```
kubectl exec eks-connector-0 --container connector-agent -n eks-connector -- cat /
var/log/amazon/ssm/amazon-ssm-agent.log
kubectl exec eks-connector-1 --container connector-agent -n eks-connector -- cat /
var/log/amazon/ssm/amazon-ssm-agent.log
```

Dapatkan nama cluster yang efektif

Cluster Amazon EKS diidentifikasi secara unik oleh `clusterName` dalam satu AWS akun dan Wilayah. AWS Jika Anda memiliki beberapa cluster yang terhubung di Amazon EKS, Anda dapat mengonfirmasi klaster Amazon EKS mana yang terdaftar untuk klaster Kubernetes saat ini. Untuk melakukan ini, masukkan yang berikut ini untuk mengetahui cluster saat ini. `clusterName`

```
kubectl exec eks-connector-0 --container connector-agent -n eks-connector \
-- cat /var/log/amazon/ssm/amazon-ssm-agent.log | grep -m1 -oE "eks_c:[a-zA-Z0-9_-]+"
| sed -E "s/^.*eks_c:([a-zA-Z0-9_-]+)_[a-zA-Z0-9]+.*$/\1/"
kubectl exec eks-connector-1 --container connector-agent -n eks-connector \
-- cat /var/log/amazon/ssm/amazon-ssm-agent.log | grep -m1 -oE "eks_c:[a-zA-Z0-9_-]+"
| sed -E "s/^.*eks_c:([a-zA-Z0-9_-]+)_[a-zA-Z0-9]+.*$/\1/"
```

Perintah lain-lain

Perintah berikut berguna untuk mengambil informasi yang Anda butuhkan untuk memecahkan masalah.

- Gunakan perintah berikut untuk mengumpulkan gambar yang digunakan oleh Pod di Amazon EKS Connector.

```
kubectl get pods -n eks-connector -o jsonpath="{.items[*].spec.containers[*].image}"
| tr -s '[:space:]' '\n'
```

- Gunakan perintah berikut untuk menentukan nama node tempat Amazon EKS Connector berjalan.

```
kubectl get pods -n eks-connector -o jsonpath="{.items[*].spec.nodeName}" | tr -s '[:space:]' '\n'
```

- Jalankan perintah berikut untuk mendapatkan versi klien dan server Kubernetes Anda.

```
kubectl version
```

- Jalankan perintah berikut untuk mendapatkan informasi tentang node Anda.

```
kubectl get nodes -o wide --show-labels
```

Masalah helm: 403 Terlarang

Jika Anda menerima kesalahan berikut saat menjalankan perintah helm install:

```
Error: INSTALLATION FAILED: unexpected status from HEAD request to https://public.ecr.aws/v2/eks-connector/eks-connector-chart/manifests/0.0.6: 403 Forbidden
```

Anda dapat menjalankan baris berikut untuk memperbaikinya:

```
docker logout public.ecr.aws
```

Kesalahan konsol: cluster macet dalam status Tertunda

Jika klaster macet dalam Pending status di konsol Amazon EKS setelah Anda mendaftarkannya, itu mungkin karena Konektor Amazon EKS AWS belum berhasil menghubungkan cluster. Untuk klaster terdaftar, Pending status berarti bahwa koneksi tidak berhasil dibuat. Untuk mengatasi masalah ini, pastikan Anda telah menerapkan manifes ke cluster Kubernetes target. Jika Anda menerapkannya ke cluster, tetapi cluster masih dalam Pending status, maka eks-connector statefulset mungkin tidak sehat. Untuk memecahkan masalah ini, lihat [the section called “Pod konektor Amazon EKS sedang crash looping”](#) di topik ini.

Kesalahan konsol: Sistem pengguna: serviceaccount:eks-connector:eks-connector tidak dapat menyamar sebagai pengguna sumber daya dalam grup API pada cakupan cluster

[Konektor Amazon EKS menggunakan peniruan identitas pengguna Kubernetes untuk bertindak atas nama kepala sekolah IAM dari.](#) AWS Management Console Setiap prinsipal yang mengakses API Kubernetes dari akun AWS eks-connector layanan harus diberikan izin untuk meniru pengguna Kubernetes yang sesuai dengan IAM ARN sebagai nama pengguna Kubernetes. Dalam contoh berikut, IAM ARN dipetakan ke pengguna Kubernetes.

- Pengguna IAM *john* dari AWS akun *111122223333* dipetakan ke pengguna Kubernetes. [Praktik terbaik IAM](#) menyarankan agar Anda memberikan izin untuk peran, bukan pengguna.

```
arn:aws:iam::111122223333:user/john
```

- Peran IAM *admin* dari AWS akun *111122223333* dipetakan ke pengguna Kubernetes:

```
arn:aws:iam::111122223333:role/admin
```

Hasilnya adalah peran IAM ARN, bukan ARN sesi AWS STS.

Untuk petunjuk tentang cara mengonfigurasi ClusterRole dan ClusterRoleBinding memberikan hak istimewa akun eks-connector layanan untuk meniru pengguna yang dipetakan, lihat. [the section called “Berikan akses ke cluster”](#) Pastikan bahwa dalam template, %IAM_ARN% diganti dengan IAM ARN dari kepala IAM AWS Management Console .

Kesalahan konsol: [...] dilarang: Pengguna [...] tidak dapat mencantumkan sumber daya [...] dalam grup API di cakupan cluster

Pertimbangkan masalah berikut. Konektor Amazon EKS telah berhasil menyamar sebagai prinsipal AWS Management Console IAM yang meminta di cluster Kubernetes target. Namun, prinsipal yang ditiru tidak memiliki izin RBAC untuk operasi API Kubernetes.

Untuk mengatasi masalah ini, ada dua metode untuk memberikan izin kepada pengguna tambahan. Jika sebelumnya Anda menginstal eks-konektor melalui bagan helm, Anda dapat dengan mudah memberikan akses kepada pengguna dengan menjalankan perintah berikut. Ganti `userARN1`

dan `userARN2` dengan daftar peran IAM untuk memberikan akses untuk melihat sumber daya Kubernetes: ARNs

```
helm upgrade eks-connector oci://public.ecr.aws/eks-connector/eks-connector-chart \
  --reuse-values \
  --set 'authentication.allowedUserARNs={userARN1,userARN2}'
```

Atau, sebagai administrator klaster, berikan tingkat hak istimewa RBAC yang sesuai kepada masing-masing pengguna Kubernetes. Untuk informasi selengkapnya dan contoh tambahan, lihat [the section called “Berikan akses ke cluster”](#).

Kesalahan konsol: Amazon EKS tidak dapat berkomunikasi dengan server API cluster Kubernetes Anda. Cluster harus dalam keadaan AKTIF agar koneksi berhasil. Coba lagi dalam beberapa menit.

Jika layanan Amazon EKS tidak dapat berkomunikasi dengan konektor Amazon EKS di cluster target, itu mungkin karena salah satu alasan berikut:

- Konektor Amazon EKS di cluster target tidak sehat.
- Konektivitas yang buruk atau koneksi yang terputus antara cluster target dan AWS Wilayah.

Untuk mengatasi masalah ini, periksa [log Amazon EKS Connector](#). Jika Anda tidak melihat kesalahan untuk Konektor Amazon EKS, coba lagi koneksi setelah beberapa menit. Jika Anda secara teratur mengalami latensi tinggi atau konektivitas intermiten untuk kluster target, pertimbangkan untuk mendaftarkan ulang cluster ke AWS Wilayah yang terletak lebih dekat dengan Anda.

Pod konektor Amazon EKS sedang crash looping

Ada banyak alasan yang dapat menyebabkan Pod konektor Amazon EKS memasuki `CrashLoopBackOff` status. Masalah ini kemungkinan melibatkan `connector-init` wadah. Periksa status Pod konektor Amazon EKS.

```
kubectl get pods -n eks-connector
```

Contoh output adalah sebagai berikut.

NAME	READY	STATUS	RESTARTS	AGE

```
eks-connector-0 0/2 Init:CrashLoopBackOff 1 7s
```

Jika output Anda mirip dengan output sebelumnya, lihat [the section called “Periksa log Konektor Amazon EKS”](#) untuk memecahkan masalah.

Gagal memulai eks-konektor: InvalidActivation

Saat Anda memulai Konektor Amazon EKS untuk pertama kalinya, ia mendaftarkan `activationId` dan `activationCode` dengan Amazon Web Services. Pendaftaran mungkin gagal, yang dapat menyebabkan `connector-init` penampung mogok dengan kesalahan yang mirip dengan kesalahan berikut.

```
F1116 20:30:47.261469 1 init.go:43] failed to initiate eks-connector:
InvalidActivation:
```

Untuk memecahkan masalah ini, pertimbangkan penyebab berikut dan perbaikan yang disarankan:

- Pendaftaran mungkin gagal karena `activationId` dan `activationCode` tidak ada dalam file manifes Anda. Jika ini masalahnya, pastikan bahwa itu adalah nilai yang benar yang dikembalikan dari operasi `RegisterCluster` API, dan itu `activationCode` ada di file manifes. `activationCode` itu ditambahkan ke rahasia Kubernetes, jadi harus dikodekan. base64 Untuk informasi selengkapnya, lihat [the section called “Langkah 1: Mendaftarkan cluster”](#).
- Pendaftaran mungkin gagal karena aktivasi Anda kedaluwarsa. Ini karena, untuk alasan keamanan, Anda harus mengaktifkan Konektor Amazon EKS dalam waktu tiga hari setelah mendaftarkan cluster. Untuk mengatasi masalah ini, pastikan manifes Amazon EKS Connector diterapkan ke cluster Kubernetes target sebelum tanggal dan waktu kedaluwarsa. Untuk mengonfirmasi tanggal kedaluwarsa aktivasi Anda, hubungi operasi `DescribeCluster` API.

```
aws eks describe-cluster --name my-cluster
```

Dalam contoh respons berikut, tanggal kedaluwarsa dan waktu dicatat sebagai `2021-11-12T22:28:51.101000-08:00`.

```
{
  "cluster": {
    "name": "my-cluster",
    "arn": "arn:aws:eks:region:11112223333:cluster/my-cluster",
    "createdAt": "2021-11-09T22:28:51.449000-08:00",
```



```

    "status": "FAILED",
    "tags": {
    },
    "connectorConfig": {
      "activationId": "000000000-0000-0000-0000-000000000000",
      "activationExpiry": "2021-11-12T22:28:51.101000-08:00",
      "provider": "OTHER",
      "roleArn": "arn:aws:iam::111122223333:role/my-connector-role"
    }
  }
}

```

Jika `activationExpiry` lulus, deregister cluster dan daftarkan lagi. Melakukan hal ini menghasilkan aktivasi baru.

Node cluster tidak memiliki konektivitas keluar

Agar berfungsi dengan baik, Konektor Amazon EKS memerlukan konektivitas keluar ke beberapa titik AWS akhir. Anda tidak dapat menghubungkan klaster pribadi tanpa konektivitas keluar ke AWS Wilayah target. Untuk mengatasi masalah ini, Anda harus menambahkan konektivitas keluar yang diperlukan. Untuk informasi tentang persyaratan konektor, lihat [the section called “Pertimbangan Amazon EKS Connector”](#).

Pod konektor Amazon EKS sedang dalam **ImagePullBackOff** keadaan

Jika Anda menjalankan `get pods` perintah dan Pod berada dalam `ImagePullBackOff` status, mereka tidak dapat berfungsi dengan baik. Jika Pod Konektor Amazon EKS dalam `ImagePullBackOff` status, mereka tidak dapat berfungsi dengan baik. Periksa status Pod Konektor Amazon EKS Anda.

```
kubectl get pods -n eks-connector
```

Contoh output adalah sebagai berikut.

NAME	READY	STATUS	RESTARTS	AGE
eks-connector-0	0/2	Init:ImagePullBackOff	0	4s

File manifes Konektor Amazon EKS default mereferensikan gambar dari [Galeri Publik Amazon ECR](#). Mungkin saja cluster Kubernetes target tidak dapat menarik gambar dari Galeri Publik Amazon ECR.

Baik menyelesaikan masalah tarik gambar Galeri Publik Amazon ECR, atau pertimbangkan untuk mencerminkan gambar di registri penampung pribadi pilihan Anda.

AWS Konektor pertanyaan yang sering diajukan

T: Bagaimana cara kerja teknologi yang mendasari di balik Konektor Amazon EKS?

J: Konektor Amazon EKS didasarkan pada agen AWS Systems Manager (Systems Manager). Konektor Amazon EKS berjalan sebagai a `StatefulSet` pada cluster Kubernetes Anda. Ini membuat koneksi dan proxy komunikasi antara server API cluster Anda dan Amazon Web Services. Ini dilakukan untuk menampilkan data cluster di konsol Amazon EKS hingga Anda memutuskan sambungan cluster dari AWS. Agen Systems Manager adalah proyek open source. Untuk informasi selengkapnya tentang proyek ini, lihat [halaman GitHub proyek](#).

T: Saya memiliki cluster Kubernetes lokal yang ingin saya sambungkan. Apakah saya perlu membuka port firewall untuk menghubungkannya?

J: Tidak, Anda tidak perlu membuka port firewall apa pun. Cluster Kubernetes hanya membutuhkan koneksi keluar ke Regions. AWS AWS layanan tidak pernah mengakses sumber daya di jaringan lokal Anda. Konektor Amazon EKS berjalan di cluster Anda dan memulai koneksi ke AWS. Ketika pendaftaran klaster selesai, AWS hanya mengeluarkan perintah ke Konektor Amazon EKS setelah Anda memulai tindakan dari konsol Amazon EKS yang memerlukan informasi dari server Kubernetes API di klaster Anda.

T: Data apa yang dikirim dari cluster saya AWS oleh Konektor Amazon EKS?

J: Konektor Amazon EKS mengirimkan informasi teknis yang diperlukan agar klaster Anda dapat didaftarkan AWS. Ini juga mengirimkan metadata cluster dan beban kerja untuk fitur konsol Amazon EKS yang diminta pelanggan. Konektor Amazon EKS hanya mengumpulkan atau mengirimkan data ini jika Anda memulai tindakan dari konsol Amazon EKS atau Amazon EKS API yang mengharuskan data dikirim. AWS Selain nomor versi Kubernetes, AWS tidak menyimpan data apa pun secara default. Ini menyimpan data hanya jika Anda mengotorisasikannya.

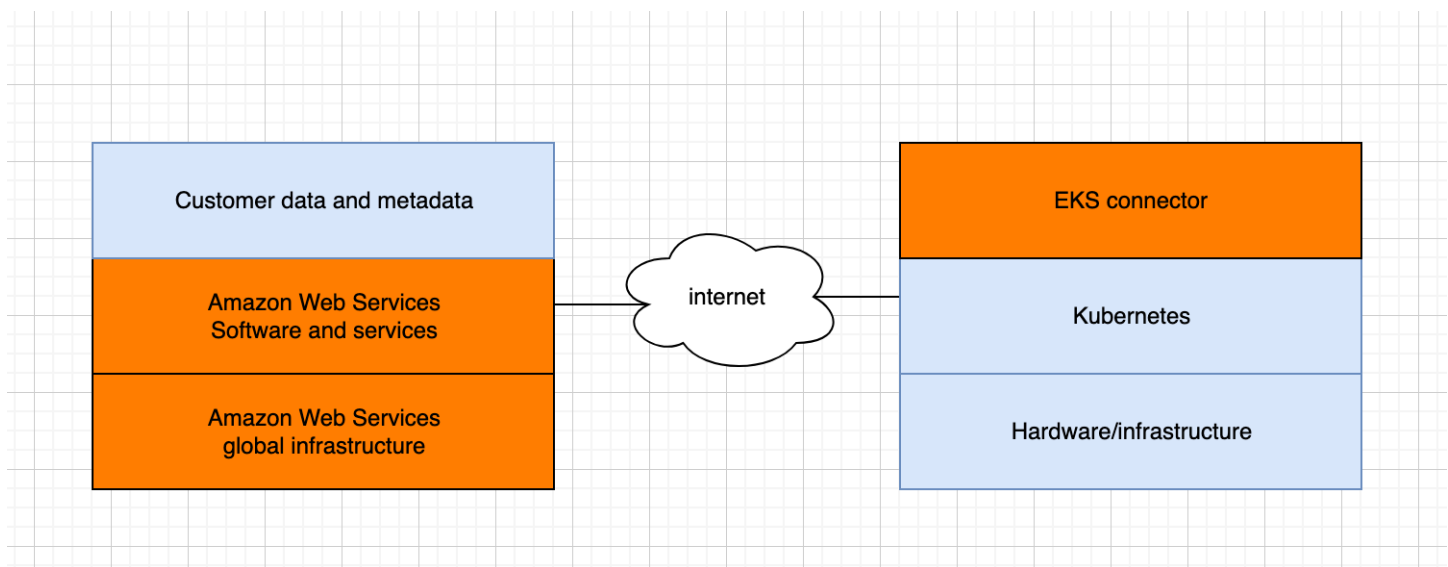
T: Dapatkah saya menghubungkan cluster di luar AWS Region?

J: Ya, Anda dapat menghubungkan cluster dari lokasi mana pun ke Amazon EKS. Selain itu, layanan Amazon EKS Anda dapat ditemukan di AWS Wilayah komersial AWS publik mana pun. Ini berfungsi dengan koneksi jaringan yang valid dari cluster Anda ke AWS Wilayah target. Sebaiknya pilih AWS Region yang paling dekat dengan lokasi klaster untuk optimasi performa UI. Misalnya, jika Anda memiliki klaster yang berjalan di Tokyo, sambungkan klaster Anda ke AWS Wilayah di Tokyo (yaitu,

ap-northeast-1 AWS Wilayah) untuk latensi rendah. Anda dapat menghubungkan kluster dari lokasi mana pun ke Amazon EKS di AWS Wilayah komersial publik mana pun, kecuali Tiongkok GovCloud AWS atau Wilayah.

Memahami keamanan di Amazon EKS Connector

Amazon EKS Connector adalah komponen open source yang berjalan di cluster Kubernetes Anda. Cluster ini dapat ditempatkan di luar AWS lingkungan. Ini menciptakan pertimbangan tambahan untuk tanggung jawab keamanan. Konfigurasi ini dapat diilustrasikan dengan diagram berikut. Oranye mewakili AWS tanggung jawab, dan biru mewakili tanggung jawab pelanggan:



Topik ini menjelaskan perbedaan model tanggung jawab jika cluster yang terhubung berada di luar AWS.

AWS tanggung jawab

- Memelihara, membangun, dan mengirimkan Amazon EKS Connector, yang merupakan [komponen open source](#) yang berjalan pada cluster Kubernetes pelanggan dan berkomunikasi dengannya. AWS
- Menjaga keamanan komunikasi lapisan transport dan aplikasi antara kluster Kubernetes dan layanan yang terhubung. AWS

Tanggung jawab pelanggan

- Keamanan khusus kluster Kubernetes, khususnya di sepanjang baris berikut:

- Rahasia Kubernetes harus dienkripsi dan dilindungi dengan benar.
- Kunci akses ke eks-connector namespace.
- [Mengkonfigurasi izin kontrol akses berbasis peran \(RBAC\) untuk mengelola akses utama IAM dari AWS](#) Untuk petunjuk, silakan lihat [the section called “Berikan akses ke cluster”](#).
- Menginstal dan meningkatkan Konektor Amazon EKS.
- Memelihara perangkat keras, perangkat lunak, dan infrastruktur yang mendukung kluster Kubernetes yang terhubung.
- [Mengamankan AWS akun mereka \(misalnya, dengan menjaga kredensi pengguna root Anda\)](#).

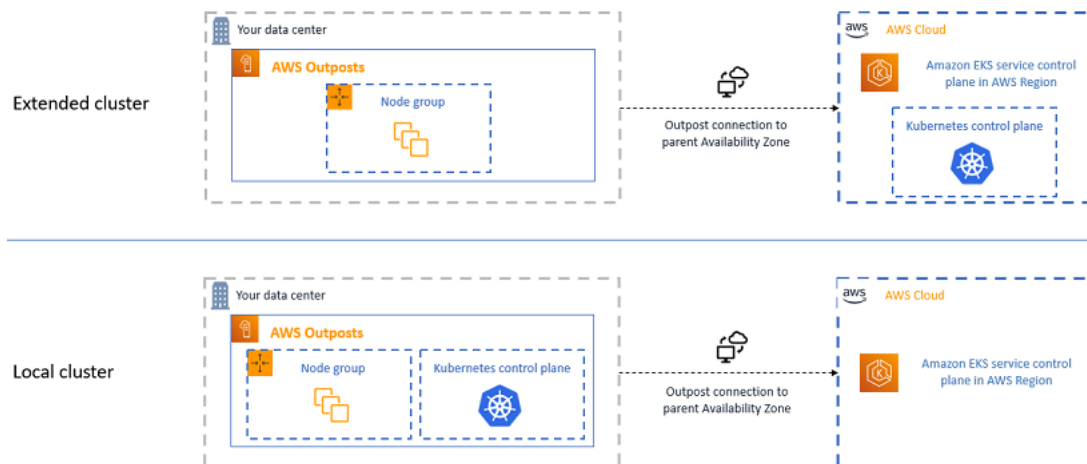
Menerapkan Amazon EKS lokal dengan Outposts AWS

Anda dapat menggunakan Amazon EKS untuk menjalankan aplikasi Kubernetes lokal di Outposts. AWS Anda dapat menerapkan Amazon EKS di Outposts dengan cara berikut:

- Extended cluster — Jalankan control plane Kubernetes di AWS Region dan node di Outpost Anda.
- Cluster lokal — Jalankan control plane Kubernetes dan node di Outpost Anda.

Untuk kedua opsi penerapan, pesawat kontrol Kubernetes dikelola sepenuhnya oleh AWS. Anda dapat menggunakan Amazon EKS APIs, alat, dan konsol yang sama yang Anda gunakan di cloud untuk membuat dan menjalankan Amazon EKS di Outposts.

Diagram berikut menunjukkan opsi penyebaran ini.



Kapan menggunakan setiap opsi penerapan

Cluster lokal dan diperluas adalah opsi penyebaran tujuan umum dan dapat digunakan untuk berbagai aplikasi.

Dengan kluster lokal, Anda dapat menjalankan seluruh kluster Amazon EKS secara lokal di Outposts. Opsi ini dapat mengurangi risiko downtime aplikasi yang mungkin diakibatkan oleh pemutusan jaringan sementara ke cloud. Pemutusan jaringan ini dapat disebabkan oleh pemotongan serat atau peristiwa cuaca. Karena seluruh kluster Amazon EKS berjalan secara lokal di Outposts, aplikasi tetap tersedia. Anda dapat melakukan operasi cluster selama pemutusan jaringan ke cloud. Untuk informasi selengkapnya, lihat [the section called “Bersiaplah untuk terputus”](#). Jika Anda khawatir

tentang kualitas koneksi jaringan dari Outposts Anda ke AWS Wilayah induk dan memerlukan ketersediaan tinggi melalui pemutusan jaringan, gunakan opsi penyebaran cluster lokal.

Dengan klaster yang diperluas, Anda dapat menghemat kapasitas di Pos Luar karena bidang kontrol Kubernetes berjalan di Wilayah induk. AWS Opsi ini cocok jika Anda dapat berinvestasi dalam konektivitas jaringan yang andal dan redundan dari Pos Luar Anda ke Wilayah. AWS Kualitas koneksi jaringan sangat penting untuk opsi ini. Cara Kubernetes menangani pemutusan jaringan antara bidang kontrol Kubernetes dan node dapat menyebabkan downtime aplikasi. Untuk informasi selengkapnya tentang perilaku Kubernetes, lihat [Penjadwalan, Preemption, dan Penggusuran di dokumentasi Kubernetes](#).

Membandingkan opsi penerapan

Tabel berikut membandingkan perbedaan antara dua opsi.

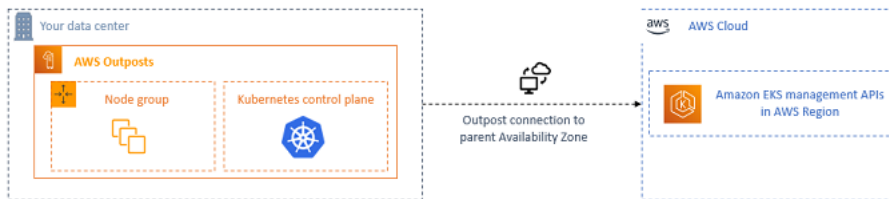
Fitur	Cluster yang diperluas	Cluster lokal
Kubernetes mengontrol lokasi pesawat	AWS Wilayah	Pos terdepan
Akun pesawat kontrol Kubernetes	AWS akun	Akun Anda
Ketersediaan wilayah	lihat Titik akhir layanan	AS Timur (Ohio), AS Timur (Virginia N.), AS Barat (California N.), AS Barat (Oregon), Asia Pasifik (Seoul), Asia Pasifik (Singapura), Asia Pasifik (Sydney), Asia Pasifik (Tokyo), Kanada (Tengah), Eropa (Frankfurt), Eropa (Irlandia), Eropa (London), Timur Tengah (Bahrain), dan Amerika Selatan (Sao Paulo)
Kubernetes versi minor	Versi Amazon EKS yang didukung.	Versi Amazon EKS yang didukung.

Fitur	Cluster yang diperluas	Cluster lokal
Versi platform	Lihat the section called “Versi platform”	Lihat the section called “Versi platform EKS”
Faktor bentuk pos terdepan	Rak pos terdepan	Rak pos terdepan
Antarmuka pengguna	AWS Management Console, AWS CLI, Amazon EKS API,, eksctl AWS CloudFormation, dan Terraform	AWS Management Console, AWS CLI, Amazon EKS API,, eksctl AWS CloudFormation, dan Terraform
Kebijakan terkelola	EKSClusterKebijakan Amazon dan the section called “AWS kebijakan terkelola: Amazon EKSService RolePolicy”	Amazon EKSLocal OutpostClusterPolicy dan the section called “AWS kebijakan terkelola: Amazon EKSLocal OutpostServiceRolePolicy”
Cluster VPC dan subnet	Lihat the section called “Persyaratan VPC dan subnet”	Lihat the section called “Buat VPC dan subnet”
Akses titik akhir kluster	Publik atau swasta atau keduanya	Hanya pribadi
Autentikasi server API Kubernetes	AWS Identity and Access Management (IAM) dan OIDC	IAM dan sertifikat x.509
Jenis simpul	Hanya dikelola sendiri	Hanya dikelola sendiri
Jenis komputasi simpul	Amazon EC2 sesuai permintaan	Amazon EC2 sesuai permintaan
Jenis penyimpanan simpul	Amazon EBS gp2 dan SSD lokal NVMe	Amazon EBS gp2 dan SSD lokal NVMe
Amazon EKS dioptimalkan AMIs	Amazon Linux, Windows, dan Bottlerocket	Hanya Amazon Linux
Versi IP	IPv4hanya	IPv4hanya

Fitur	Cluster yang diperluas	Cluster lokal
Pengaya	Pengaya Amazon EKS atau add-on yang dikelola sendiri	Hanya add-on yang dikelola sendiri
Antarmuka Jaringan Kontainer Default	Plugin Amazon VPC CNI untuk Kubernetes	Plugin Amazon VPC CNI untuk Kubernetes
Log pesawat kontrol Kubernetes	CloudWatch Log Amazon	CloudWatch Log Amazon
Penyeimbangan beban	Gunakan AWS Load Balancer Controller untuk menyediakan Application Load Balancer saja (tidak ada Network Load Balancer)	Gunakan AWS Load Balancer Controller untuk menyediakan Application Load Balancer saja (tidak ada Network Load Balancer)
Rahasia enkripsi amplop	Lihat the section called “Aktifkan enkripsi rahasia”	Tidak didukung
IAM role untuk akun layanan	Lihat the section called “Kredensial dengan IRSA”	Tidak didukung
Pemecahan Masalah	Lihat Pemecahan Masalah	Lihat the section called “Memecahkan masalah cluster”

Buat cluster Amazon EKS lokal di AWS Outposts untuk ketersediaan tinggi

Anda dapat menggunakan kluster lokal untuk menjalankan seluruh kluster Amazon EKS secara lokal di AWS Outposts. Ini membantu mengurangi risiko downtime aplikasi yang mungkin diakibatkan oleh pemutusan jaringan sementara ke cloud. Pemutusan ini dapat disebabkan oleh pemotongan serat atau peristiwa cuaca. Karena seluruh kluster Kubernetes berjalan secara lokal di Outposts, aplikasi tetap tersedia. Anda dapat melakukan operasi cluster selama pemutusan jaringan ke cloud. Untuk informasi selengkapnya, lihat [the section called “Bersiaplah untuk terputus”](#). Diagram berikut menunjukkan penyebaran cluster lokal.



Cluster lokal umumnya tersedia untuk digunakan dengan rak Outposts.

AWS Wilayah yang Didukung

Anda dapat membuat cluster lokal di AWS Wilayah berikut: AS Timur (Ohio), AS Timur (Virginia N.), AS Barat (California N.), AS Barat (Oregon), Asia Pasifik (Seoul), Asia Pasifik (Singapura), Asia Pasifik (Sydney), Asia Pasifik (Tokyo), Kanada (Tengah), Eropa (Frankfurt), Eropa (Irlandia), Eropa (London), Timur Tengah (Bahrain), dan Amerika Selatan (São Paulo). Untuk informasi rinci tentang fitur yang didukung, lihat [the section called “Membandingkan opsi penerapan”](#).

Menerapkan cluster Amazon EKS di Outposts AWS

Topik ini memberikan gambaran umum tentang apa yang harus dipertimbangkan saat menjalankan cluster lokal di Outpost. Topik ini juga memberikan instruksi tentang cara menyebarkan cluster lokal di Outpost.

⚠ Important

- Pertimbangan ini tidak direplikasi dalam dokumentasi Amazon EKS terkait. Jika topik dokumentasi Amazon EKS lainnya bertentangan dengan pertimbangan di sini, ikuti pertimbangannya di sini.
 - Pertimbangan ini dapat berubah dan mungkin sering berubah. Jadi, kami sarankan Anda meninjau topik ini secara teratur.
 - Banyak pertimbangan yang berbeda dari pertimbangan untuk membuat cluster di Cloud. AWS
-
- Cluster lokal hanya mendukung rak Outpost. Sebuah cluster lokal tunggal dapat berjalan di beberapa rak Outpost fisik yang terdiri dari satu pos logis. Satu cluster lokal tidak dapat berjalan di beberapa Outposts logis. Setiap Outpost logis memiliki ARN Outpost tunggal.
 - Cluster lokal menjalankan dan mengelola control plane Kubernetes di akun Anda di Outpost. Anda tidak dapat menjalankan beban kerja pada instance control plane Kubernetes atau memodifikasi

komponen control plane Kubernetes. Node ini dikelola oleh layanan Amazon EKS. Perubahan pada bidang kontrol Kubernetes tidak bertahan melalui tindakan manajemen Amazon EKS otomatis, seperti patching.

- Cluster lokal mendukung add-on yang dikelola sendiri dan grup node Amazon Linux yang dikelola sendiri. [Plugin Amazon VPC CNI untuk Kubernetes, kube-proxy, dan CoreDNS add-on secara otomatis diinstal pada cluster lokal.](#)
- Cluster lokal memerlukan penggunaan Amazon EBS di Outposts. Pos Luar Anda harus memiliki Amazon EBS yang tersedia untuk penyimpanan pesawat kontrol Kubernetes.
- Cluster lokal menggunakan Amazon EBS di Outposts. Pos Luar Anda harus memiliki Amazon EBS yang tersedia untuk penyimpanan pesawat kontrol Kubernetes. Outposts hanya mendukung volume Amazon EBSgp2.
- Kubernetes yang didukung Amazon EBS PersistentVolumes didukung menggunakan driver Amazon EBS CSI.
- Contoh bidang kontrol cluster lokal diatur dalam topologi [bertumpuk yang sangat tersedia](#). Dua dari tiga contoh bidang kontrol harus sehat setiap saat untuk mempertahankan kuorum. Jika kuorum hilang, hubungi AWS dukungan, karena beberapa tindakan sisi layanan akan diperlukan untuk mengaktifkan instance terkelola yang baru.

Prasyarat

- [Keakraban dengan opsi penyebaran Outposts, Pilih jenis instans, dan grup penempatan untuk kluster Amazon EKS di AWS Outposts berdasarkan pertimbangan kapasitas, serta persyaratan serta pertimbangan VPC.](#)
- Pos terdepan yang ada. Untuk informasi selengkapnya, lihat [Apa itu AWS Outposts](#).
- Alat baris `kubectl` perintah diinstal pada komputer Anda atau AWS CloudShell. Versinya bisa sama dengan atau hingga satu versi minor lebih awal atau lebih lambat dari versi Kubernetes dari kluster Anda. Misalnya, jika versi cluster Anda `1.29`, Anda dapat menggunakan `kubectl` versi `1.28`, `1.29`, atau `1.30` dengan itu. Untuk menginstal atau memutakhirkan `kubectl`, lihat [the section called “Siapkan kubectl dan eksctl”](#).
- Versi `2.12.3` atau yang lebih baru atau versi `1.27.160` atau yang lebih baru dari AWS Command Line Interface (AWS CLI) diinstal dan dikonfigurasi pada perangkat Anda atau AWS CloudShell. Untuk memeriksa versi Anda saat ini, gunakan `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Package manager seperti `yum` atau `apt-get`, atau Homebrew untuk macOS seringkali merupakan beberapa versi di belakang versi terbaru CLI. AWS Untuk menginstal versi terbaru, lihat [Menginstal](#) dan [Konfigurasi cepat dengan aws configure](#) di Panduan Pengguna Antarmuka

Baris AWS Perintah. Versi AWS CLI yang diinstal AWS CloudShell mungkin juga beberapa versi di belakang versi terbaru. Untuk memperbaruinya, lihat [Menginstal AWS CLI ke direktori home Anda](#) di AWS CloudShell Panduan Pengguna.

- Prinsipal IAM (pengguna atau peran) dengan izin `create` dan `describe` Amazon EKS. Untuk informasi selengkapnya, lihat [the section called “Buat kluster Kubernetes lokal di Outpost”](#) dan [the section called “Buat daftar atau deskripsikan semua kluster”](#).

Saat kluster Amazon EKS lokal dibuat, [prinsipal IAM](#) yang membuat cluster ditambahkan secara permanen. Prinsipal secara khusus ditambahkan ke tabel otorisasi Kubernetes RBAC sebagai administrator. Entitas ini memiliki `system:masters` izin. Identitas entitas ini tidak terlihat dalam konfigurasi kluster Anda. Jadi, penting untuk mencatat entitas yang membuat cluster dan pastikan Anda tidak pernah menghapusnya. Awalnya, hanya prinsipal yang membuat server yang dapat melakukan panggilan ke server API Kubernetes menggunakan `kubectl`. Jika Anda menggunakan konsol untuk membuat kluster, pastikan kredensial IAM yang sama ada di rantai kredensi AWS SDK saat Anda menjalankan `kubectl` perintah di kluster. Setelah cluster Anda dibuat, Anda dapat memberikan prinsipal IAM lainnya akses ke cluster Anda.

Buat kluster lokal Amazon EKS

Anda dapat membuat kluster lokal dengan alat berikut yang dijelaskan di halaman ini:

- [the section called “eksctl”](#)
- [the section called “AWS Management Console”](#)

Anda juga dapat menggunakan [AWS CLI](#), [Amazon EKS API](#), the [AWS SDKs](#), [AWS CloudFormation](#) atau [Terraform](#) untuk membuat cluster di Outposts.

eksctl

Untuk membuat cluster lokal dengan **eksctl**

1. Instal versi `0.210.0` atau yang lebih baru dari alat baris `eksctl` perintah di perangkat Anda atau AWS CloudShell. Untuk menginstal atau memperbaruiek`eksctl`, lihat [Instalasi](#) dalam `eksctl` dokumentasi.
2. Salin konten yang mengikuti ke perangkat Anda. Ganti nilai-nilai berikut dan kemudian jalankan perintah yang dimodifikasi untuk membuat `outpost-control-plane.yaml` file:
 - Ganti *region-code* dengan [AWS Wilayah yang didukung](#) tempat Anda ingin membuat kluster.

- Ganti *my-cluster* dengan nama untuk cluster Anda. Nama hanya dapat berisi karakter alfanumerik (peka huruf besar/kecil) dan tanda hubung. Itu harus dimulai dengan karakter alfanumerik dan tidak boleh lebih dari 100 karakter. Nama harus unik di dalam AWS Wilayah dan AWS akun tempat Anda membuat klaster. Nama harus unik di dalam AWS Wilayah dan AWS akun tempat Anda membuat klaster.
- Ganti *vpc-ExampleID1* dan *subnet-ExampleID1* dengan VPC dan subnet Anda yang ada. IDs VPC dan subnet harus memenuhi persyaratan di [Buat VPC dan subnet untuk cluster Amazon EKS di Outposts. AWS](#)
- Ganti *uniqueid* dengan ID Outpost Anda.
- Ganti *m5.large* dengan jenis instance yang tersedia di Outpost Anda. Sebelum memilih jenis instance, lihat [the section called "Pertimbangan kapasitas"](#). Tiga instance pesawat kontrol dikerahkan. Anda tidak dapat mengubah nomor ini.

```
cat >outpost-control-plane.yaml <<EOF
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code
  version: "1.33"

vpc:
  clusterEndpoints:
    privateAccess: true
  id: "vpc-vpc-ExampleID1"
  subnets:
    private:
      outpost-subnet-1:
        id: "subnet-subnet-ExampleID1"

outpost:
  controlPlaneOutpostARN: arn:aws:outposts:region-code:111122223333:outpost/op-uniqueid
  controlPlaneInstanceType: m5.large
EOF
```

Untuk daftar lengkap semua opsi dan default yang tersedia, lihat [AWS Outposts Support](#) dan skema file [Config](#) dalam dokumentasi. eksctl

3. Buat cluster menggunakan file konfigurasi yang Anda buat pada langkah sebelumnya. `eksctl` membuat VPC dan satu subnet di Outpost Anda untuk menyebarkan cluster.

```
eksctl create cluster -f outpost-control-plane.yaml
```

Penyediaan klaster memerlukan waktu beberapa menit. Saat cluster sedang dibuat, beberapa baris output muncul. Baris terakhir output mirip dengan baris contoh berikut.

```
[#] EKS cluster "my-cluster" in "region-code" region is ready
```

Tip

Untuk melihat sebagian besar opsi yang dapat Anda tentukan saat membuat `eksctl`, gunakan `eksctl create cluster --help` perintah. Untuk melihat semua opsi yang tersedia, Anda dapat menggunakan config file. Untuk informasi selengkapnya, lihat [Menggunakan file config](#) dan [skema file config](#) di dokumentasi `eksctl`. Anda dapat menemukan [contoh file konfigurasi](#) di GitHub.

`eksctl` Perintah secara otomatis membuat [entri akses](#) untuk prinsipal IAM (pengguna atau peran) yang membuat klaster dan memberikan izin administrator utama IAM ke objek Kubernetes di klaster. Jika Anda tidak ingin pembuat klaster memiliki akses administrator ke objek Kubernetes di cluster, tambahkan teks berikut ke file konfigurasi sebelumnya: `bootstrapClusterCreatorAdminPermissions: false` (pada level yang sama dengan `metadata`, `vpc`, dan). `outpost` Jika Anda menambahkan opsi, maka setelah pembuatan klaster, Anda perlu membuat entri akses untuk setidaknya satu prinsipal IAM, atau tidak ada prinsipal IAM yang memiliki akses ke objek Kubernetes di cluster.

AWS Management Console

Untuk membuat cluster Anda dengan AWS Management Console

1. Anda memerlukan VPC dan subnet yang sudah ada yang memenuhi persyaratan Amazon EKS. Untuk informasi selengkapnya, lihat [the section called "Buat VPC dan subnet"](#).
2. Jika Anda sudah memiliki peran IAM cluster lokal, atau Anda akan membuat cluster Anda dengan `eksctl`, maka Anda dapat melewati langkah ini. Secara default, `eksctl` buat peran untuk Anda.

- a. Jalankan perintah berikut untuk membuat file JSON kebijakan kepercayaan IAM.

```
cat >eks-local-cluster-role-trust-policy.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

- b. Buat peran IAM cluster Amazon EKS. Untuk membuat peran IAM, [prinsipal IAM](#) yang membuat peran harus diberi `iam:CreateRole` tindakan (izin).

```
aws iam create-role --role-name myAmazonEKSLocalClusterRole --assume-role-policy-
document file://"eks-local-cluster-role-trust-policy.json"
```

- c. Lampirkan kebijakan terkelola Amazon EKS bernama [Amazon EKSLocal OutpostClusterPolicy](#) ke peran tersebut. Untuk melampirkan kebijakan IAM ke kepala sekolah [IAM](#), [prinsipal](#) yang melampirkan kebijakan harus diberikan salah satu tindakan IAM berikut (izin): atau. `iam:AttachUserPolicy` `iam:AttachRolePolicy`

```
aws iam attach-role-policy --policy-arn arn:aws: iam::aws:policy/
AmazonEKSLocalOutpostClusterPolicy --role-name myAmazonEKSLocalClusterRole
```

3. Buka [konsol Amazon EKS](#).
4. Di bagian atas layar konsol, pastikan Anda telah memilih [AWS Wilayah yang didukung](#).
5. Pilih Add cluster dan kemudian pilih Create.
6. Pada halaman Configure cluster, masukkan atau pilih nilai untuk bidang berikut:
 - Kubernetes mengontrol lokasi pesawat — Pilih Outposts. AWS
 - Outpost ID - Pilih ID Outpost tempat Anda ingin membuat pesawat kontrol Anda.
 - Jenis instans - Pilih jenis instans. Hanya jenis instans yang tersedia di Outpost Anda yang ditampilkan. Dalam daftar dropdown, setiap jenis instance menjelaskan berapa banyak node

yang direkomendasikan untuk jenis instance. Sebelum memilih jenis instance, lihat [the section called “Pertimbangan kapasitas”](#). Semua replika dikerahkan menggunakan jenis instance yang sama. Anda tidak dapat mengubah jenis instance setelah cluster Anda dibuat. Tiga instance pesawat kontrol dikerahkan. Anda tidak dapat mengubah nomor ini.

- **Nama** — Nama untuk klaster Anda. Itu harus unik di AWS akun Anda. Nama hanya dapat berisi karakter alfanumerik (peka huruf besar/kecil) dan tanda hubung. Itu harus dimulai dengan karakter alfanumerik dan tidak boleh lebih dari 100 karakter. Nama harus unik di dalam AWS Wilayah dan AWS akun tempat Anda membuat klaster. Nama harus unik di dalam AWS Wilayah dan AWS akun tempat Anda membuat klaster.
- **Versi Kubernetes** — Pilih versi Kubernetes yang ingin Anda gunakan untuk klaster Anda. Sebaiknya pilih versi terbaru, kecuali jika Anda perlu menggunakan versi sebelumnya.
- **Peran layanan klaster** — Pilih peran IAM klaster Amazon EKS yang Anda buat pada langkah sebelumnya untuk memungkinkan bidang kontrol Kubernetes mengelola sumber daya. AWS
- **Akses administrator klaster Kubernetes** — Jika Anda ingin prinsipal IAM (peran atau pengguna) yang membuat klaster memiliki akses administrator ke objek Kubernetes di klaster, terima default (allow). Amazon EKS membuat entri akses untuk prinsipal IAM dan memberikan izin administrator klaster ke entri akses. Untuk informasi selengkapnya tentang entri akses, lihat [the section called “Berikan izin”](#).

Jika Anda menginginkan prinsipal IAM yang berbeda dari prinsipal yang membuat klaster memiliki akses administrator ke objek cluster Kubernetes, pilih opsi disallow. Setelah pembuatan klaster, setiap prinsipal IAM yang memiliki izin IAM untuk membuat entri akses dapat menambahkan entri akses untuk setiap prinsipal IAM yang memerlukan akses ke objek klaster Kubernetes. Untuk informasi selengkapnya tentang izin IAM yang diperlukan, lihat [Tindakan yang ditentukan oleh Amazon Elastic Kubernetes Service](#) di Referensi Otorisasi Layanan. Jika Anda memilih opsi disallow dan tidak membuat entri akses apa pun, maka tidak ada prinsipal IAM yang akan memiliki akses ke objek Kubernetes di cluster.

- **Tanda** — (Opsional) Tambahkan tanda apapun ke klaster Anda. Untuk informasi selengkapnya, lihat [the section called “Pemberian tag pada sumber daya Anda”](#). Setelah selesai dengan halaman ini, pilih Berikutnya.

7. Pada halaman Tentukan jaringan, pilih nilai untuk kolom berikut:

- **VPC** — Pilih VPC yang ada. VPC harus memiliki cukup banyak alamat IP yang tersedia untuk cluster, node apa pun, dan sumber daya Kubernetes lainnya yang ingin Anda buat. VPC Anda harus memenuhi persyaratan dalam persyaratan dan pertimbangan [VPC](#).

- Subnet — Secara default, semua subnet yang tersedia di VPC yang ditentukan di bidang sebelumnya telah dipilih sebelumnya. Subnet yang Anda pilih harus memenuhi persyaratan dalam [persyaratan dan pertimbangan Subnet](#).
 - Grup keamanan — (Opsional) Tentukan satu atau beberapa grup keamanan yang ingin Anda kaitkan Amazon EKS ke antarmuka jaringan yang dibuatnya. Amazon EKS secara otomatis membuat grup keamanan yang memungkinkan komunikasi antara cluster dan VPC Anda. Amazon EKS mengaitkan grup keamanan ini, dan apa pun yang Anda pilih, ke antarmuka jaringan yang dibuatnya. Untuk informasi selengkapnya tentang grup keamanan klaster yang dibuat Amazon EKS, lihat [the section called “Persyaratan grup keamanan”](#). Anda dapat mengubah aturan di grup keamanan klaster yang dibuat Amazon EKS. Jika Anda memilih untuk menambahkan grup keamanan Anda sendiri, Anda tidak dapat mengubah grup yang Anda pilih setelah pembuatan klaster. Agar host lokal dapat berkomunikasi dengan titik akhir klaster, Anda harus mengizinkan lalu lintas masuk dari grup keamanan klaster. Untuk cluster yang tidak memiliki koneksi internet ingress dan egress (juga dikenal sebagai cluster pribadi), Anda harus melakukan salah satu hal berikut:
 - Tambahkan grup keamanan yang terkait dengan titik akhir VPC yang diperlukan. Untuk informasi selengkapnya tentang titik akhir yang diperlukan, lihat [the section called “Menggunakan VPC endpoint antarmuka ”](#) di [Subnet akses ke AWS layanan](#).
 - Ubah grup keamanan yang dibuat Amazon EKS untuk memungkinkan lalu lintas dari grup keamanan yang terkait dengan titik akhir VPC. Setelah selesai dengan halaman ini, pilih Berikutnya.
8. Pada halaman Konfigurasi observabilitas, Anda dapat secara opsional memilih opsi pencatatan bidang Metrik dan Kontrol mana yang ingin Anda aktifkan. Secara default, setiap jenis log dimatikan.
- Untuk informasi selengkapnya tentang opsi metrik Prometheus, lihat. [the section called “Langkah 1: Aktifkan metrik Prometheus”](#)
 - Untuk informasi selengkapnya tentang opsi Pencatatan bidang kontrol, lihat [the section called “Kontrol log pesawat”](#). Setelah selesai dengan halaman ini, pilih Berikutnya.
9. Pada halaman Tinjau dan buat, tinjau informasi yang Anda masukkan atau pilih pada halaman sebelumnya. Jika Anda perlu melakukan perubahan, pilih Edit. Saat Anda puas, pilih Buat. Bidang Status menunjukkan CREATING saat cluster disediakan.

Penyediaan klaster memerlukan waktu beberapa menit.

Lihat kluster lokal Amazon EKS Anda

1. Setelah cluster dibuat, Anda dapat melihat instance bidang EC2 kontrol Amazon yang dibuat.

```
aws ec2 describe-instances --query 'Reservations[*].Instances[*].{Name:Tags[?Key=='Name`']|[0].Value}' | grep my-cluster-control-plane
```

Contoh output adalah sebagai berikut.

```
"Name": "my-cluster-control-plane-id1"  
"Name": "my-cluster-control-plane-id2"  
"Name": "my-cluster-control-plane-id3"
```

Setiap instance dicemari `node-role.eks-local.amazonaws.com/control-plane` sehingga tidak ada beban kerja yang dijadwalkan pada instance bidang kontrol. Untuk informasi selengkapnya tentang taints, lihat [Taints and Tolerations](#) dalam dokumentasi Kubernetes. Amazon EKS terus memantau keadaan cluster lokal. Kami melakukan tindakan manajemen otomatis, seperti patch keamanan dan memperbaiki instans yang tidak sehat. Ketika kluster lokal terputus dari cloud, kami menyelesaikan tindakan untuk memastikan bahwa kluster diperbaiki ke keadaan sehat setelah tersambung kembali.

2. Jika Anda membuat cluster Anda menggunakan `eksctl`, maka Anda dapat melewati langkah ini. `eksctl` Selesaikan langkah ini untuk Anda. Aktifkan `kubectl` untuk berkomunikasi dengan cluster Anda dengan menambahkan konteks baru ke `kubectl` config file. Untuk petunjuk tentang cara membuat dan memperbarui file, lihat [the section called “Akses kluster dengan kubectl”](#).

```
aws eks update-kubeconfig --region region-code --name my-cluster
```

Contoh output adalah sebagai berikut.

```
Added new context arn:aws:eks:region-code:111122223333:cluster/my-cluster to /home/  
username/.kube/config
```

3. Untuk terhubung ke server API Kubernetes kluster lokal Anda, dapatkan akses ke gateway lokal untuk subnet, atau sambungkan dari dalam VPC. Untuk informasi selengkapnya tentang menghubungkan rak Outpost ke jaringan lokal, lihat [Cara kerja gateway lokal untuk rak di Panduan Pengguna Outposts. AWS](#) Jika Anda menggunakan Direct VPC Routing dan subnet Outpost memiliki rute ke gateway lokal Anda, alamat IP pribadi dari instans control plane Kubernetes secara otomatis disiarkan melalui jaringan lokal Anda. Titik akhir server Kubernetes API cluster

lokal di-host di Amazon Route 53 (Route 53). Endpoint layanan API dapat diselesaikan oleh server DNS publik ke alamat IP pribadi server Kubernetes API.

Instance bidang kontrol Kubernetes cluster lokal dikonfigurasi dengan antarmuka jaringan elastis statis dengan alamat IP pribadi tetap yang tidak berubah sepanjang siklus hidup klaster. Mesin yang berinteraksi dengan server API Kubernetes mungkin tidak memiliki konektivitas ke Route 53 selama pemutusan jaringan. Jika ini masalahnya, kami sarankan untuk mengonfigurasi `/etc/hosts` dengan alamat IP pribadi statis untuk operasi lanjutan. Kami juga merekomendasikan untuk menyiapkan server DNS lokal dan menghubungkannya ke Outpost Anda. Untuk informasi selengkapnya, lihat [AWS dokumentasi Outposts](#). Jalankan perintah berikut untuk mengonfirmasi bahwa komunikasi telah dibuat dengan cluster Anda.

```
kubectl get svc
```

Contoh output adalah sebagai berikut.

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	ClusterIP	10.100.0.1	<none>	443/TCP	28h

4. (Opsional) Uji otentikasi ke klaster lokal Anda saat berada dalam status terputus dari Cloud. AWS Untuk petunjuk, lihat [the section called “Bersiaplah untuk terputus”](#).

Sumber daya internal

Amazon EKS membuat sumber daya berikut di klaster Anda. Sumber dayanya untuk penggunaan internal Amazon EKS. Agar klaster berfungsi dengan baik, jangan mengedit atau memodifikasi sumber daya ini.

- [Pod cermin](#) berikut:
 - `aws-iam-authenticator-node-hostname`
 - `eks-certificates-controller-node-hostname`
 - `etcd-node-hostname`
 - `kube-apiserver-node-hostname`
 - `kube-controller-manager-node-hostname`
 - `kube-scheduler-node-hostname`
- Pengaya yang dikelola sendiri berikut ini:

- kube-system/coredns
- kube-system/kube-proxy(tidak dibuat sampai Anda menambahkan node pertama Anda)
- kube-system/aws-node(tidak dibuat sampai Anda menambahkan node pertama Anda). Cluster lokal menggunakan plugin Amazon VPC CNI untuk plugin Kubernetes untuk jaringan cluster. Jangan mengubah konfigurasi untuk instance control plane (Pod bernamaaws-node-controlplane-*). Ada variabel konfigurasi yang dapat Anda gunakan untuk mengubah nilai default ketika plugin membuat antarmuka jaringan baru. Untuk informasi lebih lanjut, lihat [dokumentasi](#) di GitHub.
- Layanan berikut:
 - default/kubernetes
 - kube-system/kube-dns
- Sebuah PodSecurityPolicy bernama eks.system
- Sebuah ClusterRole bernama eks:system:podsecuritypolicy
- Sebuah ClusterRoleBinding bernama eks:system
- Selain [grup keamanan cluster](#), Amazon EKS membuat grup keamanan di AWS akun Anda yang diberi namaeks-local-internal-do-not-use-or-edit-*cluster-name-uniqueid* . Grup keamanan ini memungkinkan lalu lintas mengalir bebas di antara komponen Kubernetes yang berjalan pada instance control plane.

Langkah selanjutnya yang disarankan:

- [Berikan prinsipal IAM yang membuat klaster izin yang diperlukan untuk melihat sumber daya Kubernetes di AWS Management Console](#)
- [Berikan akses entitas IAM ke klaster Anda](#). Jika Anda ingin entitas melihat resource Kubernetes di konsol Amazon EKS, berikan [izin yang Diperlukan](#) ke entitas.
- [Konfigurasi logging untuk klaster Anda](#)
- Biasakan diri Anda dengan apa yang terjadi selama [pemutusan jaringan](#).
- [Tambahkan node ke cluster Anda](#)
- Pertimbangkan untuk menyiapkan rencana cadangan untuk Andaetcd. Amazon EKS tidak mendukung pencadangan dan pemulihan otomatis etcd untuk cluster lokal. Untuk informasi selengkapnya, lihat [Mencadangkan klaster etcd](#) di dokumentasi Kubernetes. Dua opsi utama digunakan etcdctl untuk mengotomatiskan pengambilan snapshot atau menggunakan cadangan volume penyimpanan Amazon EBS.

Pelajari versi platform Kubernetes dan Amazon EKS untuk Outposts AWS

Versi platform cluster lokal mewakili kemampuan cluster Amazon EKS di AWS Outposts. Versi-versi tersebut menyertakan komponen yang berjalan pada control plane Kubernetes, dimana flag server API Kubernetes diaktifkan. Mereka juga menyertakan versi patch Kubernetes saat ini. Setiap versi minor Kubernetes memiliki satu atau lebih versi platform terkait. Versi platform untuk versi minor Kubernetes yang berbeda bersifat independen. Versi platform untuk cluster lokal dan kluster Amazon EKS di cloud bersifat independen.

Ketika versi minor Kubernetes baru tersedia untuk kluster lokal, seperti `eks-local-outposts.1.31`, versi platform awal untuk versi minor Kubernetes dimulai pada `eks-local-outposts.1`. Namun, Amazon EKS merilis versi platform baru secara berkala guna mengaktifkan pengaturan pesawat kendali Kubernetes baru dan untuk menyediakan perbaikan keamanan.

Saat versi platform cluster lokal baru tersedia untuk versi minor:

- Nomor versi platform bertambah (`eks-local-outposts.n+1`).
- Amazon EKS secara otomatis memperbarui semua cluster lokal yang ada ke versi platform terbaru untuk versi minor Kubernetes yang sesuai. Pembaruan otomatis versi platform yang ada diluncurkan secara bertahap. Proses roll-out terdiri dari penggantian instans control plane Kubernetes terkelola yang berjalan di Outpost, satu per satu, hingga ketiga instans diganti dengan yang baru.
- Proses penggantian instans pesawat kontrol Kubernetes akan berhenti berkembang jika ada risiko gangguan layanan. Amazon EKS hanya akan mencoba mengganti instance jika 2 instance control-plane Kubernetes lainnya sehat dan melewati semua kondisi kesiapan sebagai node cluster.
- Peluncuran versi platform biasanya akan memakan waktu kurang dari 30 menit untuk menyelesaikannya. Jika kluster tetap dalam `UPDATING` status untuk jangka waktu yang lama, lihat [the section called “Memecahkan masalah cluster”](#) dan cari bantuan dari AWS Support. Jangan pernah menghentikan instance control-plane Kubernetes secara manual kecuali diinstruksikan oleh Support. AWS
- Amazon EKS boleh menerbitkan AMI simpul baru dengan versi patch yang sesuai. Semua versi patch kompatibel antara bidang kontrol Kubernetes dan node AMIs untuk satu versi minor Kubernetes.

Versi platform baru tidak memperkenalkan perubahan yang melanggar atau menyebabkan gangguan layanan.

Cluster lokal selalu dibuat dengan versi platform terbaru yang tersedia (`eks-local-outposts.n`) untuk versi Kubernetes yang ditentukan.

Versi platform saat ini dan terbaru dijelaskan dalam tabel berikut.

Untuk menerima pemberitahuan semua perubahan file sumber ke halaman dokumentasi khusus ini, Anda dapat berlangganan URL berikut dengan pembaca RSS:

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/outposts/eks-outposts-platform-versions.adoc.atom
```

Versi Kubernetes 1.31

Pengontrol penerimaan berikut diaktifkan untuk semua versi 1.31

`platform:CertificateApproval,,,CertificateSigning,CertificateSubjectRestriction,ClusterTaintNodesByConditionValidatingAdmissionPolicy,`
`danValidatingAdmissionWebhook.`

Versi Kubernetes	Amazon EKS versi platform	Catatan rilis	Tanggal rilis
1.31.6	<code>eks-local-outposts.1</code>	Rilis awal versi Kubernetes untuk cluster Amazon EKS v1.31 lokal di Outposts.	April 9, 2025

Versi Kubernetes 1.30

Pengontrol penerimaan berikut diaktifkan untuk semua versi 1.30

`platform:CertificateApproval,,,CertificateSigning,CertificateSubjectRestriction,ClusterTaintNodesByConditionValidatingAdmissionPolicy,`
`danValidatingAdmissionWebhook.`

Versi Kubernetes	Amazon EKS versi platform	Catatan rilis	Tanggal rilis
1.30.10	eks-local-outposts.3	Versi platform baru dengan perbaikan dan penyempurnaan keamanan. kube-proxy diperbaru i ke. v1.30.10 AWS IAM Authentic ator diperbarui ke. v0.6.29 Plugin Amazon VPC CNI untuk Kubernete s diperbarui ke. v1.19.2 CoreDNS diperbarui ke. v1.11.4 AWS Cloud Controller Manager diperbaru i ke v1.30.8. Versi bottlerocket diperbaru i ke. v1.34.0	Maret 27, 2025
1.30.7	eks-local-outposts.2	Versi platform baru dengan perbaikan dan penyempurnaan keamanan. kube-proxy diperbaru i ke. v1.30.7 AWS IAM Authentic ator diperbarui ke. v0.6.28 Plugin Amazon VPC CNI untuk Kubernete s diperbarui ke.	Januari 10, 2025

Versi Kubernetes	Amazon EKS versi platform	Catatan rilis	Tanggal rilis
		v1.19.0 Diperbarui versi Bottlerocket ke. v1.29.0	
1.30.5	eks-local-outposts.1	Rilis awal versi Kubernetes untuk cluster Amazon EKS v1.30 lokal di Outposts.	November 13, 2024

Versi Kubernetes 1.29

Pengontrol penerimaan berikut diaktifkan untuk semua versi 1.29
platform:CertificateApproval,,,CertificateSigning,CertificateSubjectRestriction,ClusterTaintNodesByConditionValidatingAdmissionPolicy,
danValidatingAdmissionWebhook.

Versi Kubernetes	Amazon EKS versi platform	Catatan rilis	Tanggal rilis
v1.29.14	eks-local-outposts.6	Versi platform baru dengan perbaikan dan penyempurnaan keamanan. kube-proxy diperbarui ke. v1.29.14 Plugin Amazon VPC CNI untuk Kubernetes diperbarui ke. v1.19.2 CoreDNS diperbarui ke. v1.11.4 AWS Cloud Controller Manager diperbaru	Maret 27, 2025

Versi Kubernetes	Amazon EKS versi platform	Catatan rilis	Tanggal rilis
		i kev1.29.8. Versi bottlerocket diperbaru i ke. v1.34.0	
v1.29.11	eks-local-outposts.5	Versi platform baru dengan perbaikan dan penyempurnaan keamanan. kube-proxy diperbarui ke. v1.29.11 Plugin Amazon VPC CNI untuk Kubernetes diperbarui ke. v1.19.0 Diperbarui gambar CoreDNS ke. v1.11.3 Diperbarui versi Bottlerocket ke. v1.29.0	Januari 10, 2025
1.29.9	eks-local-outposts.4	Versi platform baru dengan perbaikan dan penyempurnaan keamanan. kube-proxy diperbaru i ke. v1.29.9 AWS IAM Authentic ator diperbarui ke. v0.6.26 Diperbarui versi Bottlerocket ke. v1.26.0	November 8, 2024

Versi Kubernetes	Amazon EKS versi platform	Catatan rilis	Tanggal rilis
1.29.6	eks-local-outposts.3	Versi platform baru dengan perbaikan dan penyempurnaan keamanan. Diperbarui versi Bottlerocket ke. v1.22.0	Oktober 22, 2024
1.29.6	eks-local-outposts.2	Versi platform baru dengan perbaikan dan penyempurnaan keamanan. Diperbarui versi Bottlerocket ke. v1.21.0	Agustus 27, 2024
1.29.6	eks-local-outposts.1	Rilis awal versi Kubernetes untuk cluster Amazon EKS v1.29 lokal di Outposts.	Agustus 20, 2024

Versi Kubernetes 1.28

Pengontrol penerimaan berikut diaktifkan untuk semua versi 1.28

platform:CertificateApproval,,,CertificateSigning,CertificateSubjectRestriction,ClusterTaintNodesByConditionValidatingAdmissionPolicy, danValidatingAdmissionWebhook.

Versi Kubernetes	Amazon EKS versi platform	Catatan rilis	Tanggal rilis
1.28.15	eks-local-outposts.13	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Maret 27, 2025

Versi Kubernetes	Amazon EKS versi platform	Catatan rilis	Tanggal rilis
		v1.28.15 Kube-proxy build diperbarui. i. Plugin Amazon VPC CNI untuk Kubernetes diperbarui ke. v1.19.0 AWS Cloud Controller Manager diperbarui ke v1.28.11.	
1.28.15	eks-local-outposts.12	Versi platform baru dengan perbaikan dan penyempurnaan keamanan. kube-proxy diperbarui ke. v1.28.15 Plugin Amazon VPC CNI untuk Kubernetes diperbarui ke. v1.19.0 Diperbarui versi Bottlerocket ke. v1.29.0	Januari 10, 2025

Versi Kubernetes	Amazon EKS versi platform	Catatan rilis	Tanggal rilis
1.28.14	eks-local-outposts.11	Versi platform baru dengan perbaikan dan penyempurnaan keamanan. kube-proxy diperbarui ke. v1.28.14 AWS IAM Authenticator diperbarui ke. v0.6.26 Diperbarui gambar CoreDNS ke. v1.11.1 Diperbarui versi Bottlerocket ke. v1.26.0	November 8, 2024
1.28.10	eks-local-outposts.10	Versi platform baru dengan perbaikan dan penyempurnaan keamanan. Diperbarui ke. Bottlerocket ke. v1.22.0	Oktober 22, 2024
1.28.10	eks-local-outposts.9	Versi platform baru dengan perbaikan dan penyempurnaan keamanan. Diperbarui ke. Bottlerocket ke. v1.21.0	Agustus 27, 2024
1.28.10	eks-local-outposts.8	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Juli 30, 2024

Versi Kubernetes	Amazon EKS versi platform	Catatan rilis	Tanggal rilis
1.28.10	eks-local-outposts.6	Versi platform baru dengan perbaikan dan penyempurnaan keamanan. kube-proxy diperbarui ke. v1.28.10 AWS IAM Authenticator diperbarui ke. v0.6.20 Diperbarui versi Bottlerocket ke. v1.20.2	Juni 19, 2024
1.28.6	eks-local-outposts.5	Diperbarui versi Bottlerocket ke v1.19.3 berisi perbaikan bug terbaru untuk mendukung boot lokal di Outposts.	April 18, 2024
1.28.6	eks-local-outposts.4	Versi platform baru dengan perbaikan dan penyempurnaan keamanan. Dukungan yang dipulihkan atau boot lokal di Outposts. Versi Bottlerocket yang diturunkan menjadi kompatibilitas. v1.15.1	April 2, 2024
1.28.6	eks-local-outposts.3	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Maret 22, 2024

Versi Kubernetes	Amazon EKS versi platform	Catatan rilis	Tanggal rilis
1.28.6	eks-local-outposts.2	Versi platform baru dengan perbaikan dan penyempurnaan keamanan. kube-proxy diperbarui ke v1.28.6 AWS IAM Authenticator diperbarui ke v0.6.17 Plugin Amazon VPC CNI untuk Kubernetes diturunkan ke karena alasan kompatibilitas. v1.13.2 Diperbarui versi Bottlerocket ke v1.19.2	8 Maret 2024
1.28.1	eks-local-outposts.1	Rilis awal versi Kubernetes untuk cluster Amazon EKS v1.28 lokal di Outposts.	4 Oktober 2023

Versi Kubernetes 1.27

Pengontrol penerimaan berikut diaktifkan untuk semua versi 1.27

platform:CertificateApproval,,CertificateSigning,CertificateSubjectRestriction,ClusterTaintNodesByConditionValidatingAdmissionPolicy,
danValidatingAdmissionWebhook.

Versi Kubernetes	Amazon EKS versi platform	Catatan rilis	Tanggal rilis
1.27.16	eks-local-outposts.13	Versi platform baru dengan perbaikan dan penyempurnaan keamanan. Plugin Amazon VPC CNI untuk Kubernetes diperbarui ke. v1.19.2 Versi bottlerocket diperbarui ke. v1.34.0	Maret 27, 2025
1.27.16	eks-local-outposts.12	Versi platform baru dengan perbaikan dan penyempurnaan keamanan. Plugin Amazon VPC CNI untuk Kubernetes diperbarui ke. v1.19.0 Diperbarui versi Bottlerocket ke. v1.29.0	Januari 10, 2025
1.27.16	eks-local-outposts.11	Versi platform baru dengan perbaikan dan penyempurnaan keamanan. kube-proxy diperbarui ke. v1.27.16 AWS IAM Authenticator diperbarui ke. v0.6.26 Diperbarui gambar CoreDNS ke. v1.11.1 Diperbarui	November 8, 2024

Versi Kubernetes	Amazon EKS versi platform	Catatan rilis	Tanggal rilis
		versi Bottlerocket ke. v1.26.0	
1.27.14	eks-local-outposts.10	Versi platform baru dengan perbaikan dan penyempurnaan keamanan. Diperbarui ke versi Bottlerocket ke. v1.22.0	Oktober 22, 2024
1.27.14	eks-local-outposts.9	Versi platform baru dengan perbaikan dan penyempurnaan keamanan. Diperbarui ke versi Bottlerocket ke. v1.21.0	Agustus 27, 2024
1.27.14	eks-local-outposts.8	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	Juli 30, 2024
1.27.14	eks-local-outposts.6	Versi platform baru dengan perbaikan dan penyempurnaan keamanan. kube-proxy diperbarui ke. v1.27.14 AWS IAM Authenticator diperbarui ke. v0.6.20 Diperbarui ke versi Bottlerocket ke. v1.20.2	Juni 19, 2024

Versi Kubernetes	Amazon EKS versi platform	Catatan rilis	Tanggal rilis
1.27.10	eks-local-outposts.5	Versi platform baru dengan perbaikan dan penyempurnaan keamanan.	April 2, 2024
1.27.10	eks-local-outposts.4	Versi platform baru dengan perbaikan dan penyempurnaan keamanan. kube-proxy diperbarui ke. v1.27.10 AWS IAM Authenticator diperbarui ke. v0.6.17 Diperbarui versi Bottlerocket ke. v1.19.2	Maret 22, 2024
1.27.3	eks-local-outposts.3	Versi platform baru dengan perbaikan dan penyempurnaan keamanan. kube-proxy diperbarui ke. v1.27.3 Plugin Amazon VPC CNI untuk Kubernetes diperbarui ke. v1.13.2	14 Juli 2023
1.27.1	eks-local-outposts.2	Diperbarui gambar CoreDNS ke. v1.10.1	22 Juni 2023

Versi Kubernetes	Amazon EKS versi platform	Catatan rilis	Tanggal rilis
1.27.1	eks-local-outposts.1	Rilis awal versi Kubernetes untuk cluster Amazon EKS 1.27 lokal di Outposts.	30 Mei 2023

Buat VPC dan subnet untuk cluster Amazon EKS di Outposts AWS

Saat membuat cluster lokal, Anda menentukan VPC dan setidaknya satu subnet pribadi yang berjalan di Outposts. Topik ini memberikan gambaran umum tentang persyaratan dan pertimbangan VPC dan subnet untuk klaster lokal Anda.

Persyaratan dan pertimbangan VPC

Saat Anda membuat klaster lokal, VPC yang Anda tentukan harus memenuhi persyaratan dan pertimbangan berikut:

- Pastikan VPC memiliki cukup alamat IP untuk cluster lokal, node apa pun, dan sumber daya Kubernetes lainnya yang ingin Anda buat. Jika VPC yang ingin Anda gunakan tidak memiliki cukup alamat IP, tingkatkan jumlah alamat IP yang tersedia. Anda dapat melakukan ini dengan [mengaitkan blok Classless Inter-Domain Routing \(CIDR\) tambahan](#) dengan VPC Anda. Anda dapat mengaitkan blok CIDR pribadi (RFC 1918) dan publik (non-RFC 1918) ke VPC Anda baik sebelum atau setelah Anda membuat cluster Anda. Diperlukan waktu cluster hingga 5 jam agar blok CIDR yang Anda kaitkan dengan VPC dikenali.
- VPC tidak dapat menetapkan awalan IP atau blok CIDR. IPv6 Karena kendala ini, informasi yang tercakup dalam [Menetapkan lebih banyak alamat IP ke node Amazon EKS dengan awalan](#) dan [the section called "IPv6"](#) tidak berlaku untuk VPC Anda.
- VPC memiliki nama host DNS dan resolusi DNS diaktifkan. Tanpa fitur-fitur ini, cluster lokal gagal dibuat, dan Anda perlu mengaktifkan fitur dan membuat ulang cluster Anda. Untuk informasi selengkapnya, lihat [atribut DNS untuk VPC Anda](#) di Panduan Pengguna Amazon VPC.
- Untuk mengakses kluster lokal Anda melalui jaringan lokal Anda, VPC harus dikaitkan dengan tabel rute gateway lokal Outpost Anda. Untuk informasi selengkapnya, lihat [asosiasi VPC](#) di Panduan Pengguna AWS Outposts.

Persyaratan dan pertimbangan subnet

Saat Anda membuat cluster, tentukan setidaknya satu subnet pribadi. Jika Anda menentukan lebih dari satu subnet, instance bidang kontrol Kubernetes didistribusikan secara merata di seluruh subnet. Jika lebih dari satu subnet ditentukan, subnet harus ada di Outpost yang sama. Selain itu, subnet juga harus memiliki rute yang tepat dan izin grup keamanan untuk berkomunikasi satu sama lain. Saat Anda membuat cluster lokal, subnet yang Anda tentukan harus memenuhi persyaratan berikut:

- Subnet semuanya berada di Outpost logis yang sama.
- Subnet bersama-sama memiliki setidaknya tiga alamat IP yang tersedia untuk instance control plane Kubernetes. Jika tiga subnet ditentukan, setiap subnet harus memiliki setidaknya satu alamat IP yang tersedia. Jika dua subnet ditentukan, setiap subnet harus memiliki setidaknya dua alamat IP yang tersedia. Jika satu subnet ditentukan, subnet harus memiliki setidaknya tiga alamat IP yang tersedia.
- Subnet memiliki rute ke [gateway lokal](#) rak Outpost untuk mengakses server API Kubernetes melalui jaringan lokal Anda. Jika subnet tidak memiliki rute ke gateway lokal rak Outpost, Anda harus berkomunikasi dengan server API Kubernetes Anda dari dalam VPC.
- Subnet harus menggunakan penamaan berbasis alamat IP. EC2 [Penamaan berbasis sumber daya](#) Amazon tidak didukung oleh Amazon EKS.

Akses subnet ke layanan AWS

Subnet pribadi cluster lokal di Outposts harus dapat berkomunikasi dengan AWS layanan Regional. [Anda dapat mencapai ini dengan menggunakan gateway NAT untuk akses internet keluar atau, jika Anda ingin menjaga semua lalu lintas pribadi dalam VPC Anda, menggunakan titik akhir VPC antarmuka.](#)

Menggunakan gateway NAT

Subnet pribadi cluster lokal di Outposts harus memiliki tabel rute terkait yang memiliki rute ke gateway NAT di subnet publik yang berada di Availability Zone induk Outpost. Subnet publik harus memiliki rute ke [gateway internet](#). Gateway NAT memungkinkan akses internet keluar dan mencegah koneksi masuk yang tidak diminta dari internet ke instance di Outpost.

Menggunakan VPC endpoint antarmuka

Jika subnet pribadi cluster lokal di Outposts tidak memiliki koneksi internet keluar, atau jika Anda ingin menjaga semua lalu lintas pribadi dalam VPC Anda, maka Anda harus membuat titik akhir VPC antarmuka berikut [dan titik akhir gateway](#) di subnet Regional sebelum membuat cluster Anda.

Titik Akhir	Jenis titik akhir
<code>com.amazonaws. <i>region-code</i> .ssm</code>	Antarmuka
<code>com.amazonaws. <i>region-code</i> .ssmmessages</code>	Antarmuka
<code>com.amazonaws. <i>region-code</i> .ec2messages</code>	Antarmuka
<code>com.amazonaws. <i>region-code</i> .ec2</code>	Antarmuka
<code>com.amazonaws. <i>region-code</i> .secretsmanager</code>	Antarmuka
<code>com.amazonaws. <i>region-code</i> .logs</code>	Antarmuka
<code>com.amazonaws. <i>region-code</i> .sts</code>	Antarmuka
<code>com.amazonaws. <i>region-code</i> .ecr.api</code>	Antarmuka
<code>com.amazonaws. <i>region-code</i> .ecr.dkr</code>	Antarmuka
<code>com.amazonaws. <i>region-code</i> .s3</code>	Gateway

Titik akhir harus memenuhi persyaratan berikut:

- Dibuat di subnet pribadi yang terletak di Availability Zone induk Outpost Anda
- Miliki nama DNS pribadi diaktifkan
- Memiliki grup keamanan terlampir yang memungkinkan lalu lintas HTTPS masuk dari kisaran CIDR dari subnet pos terdepan pribadi.

Membuat titik akhir menimbulkan biaya. Untuk informasi selengkapnya, lihat [harga AWS PrivateLink](#). Jika Pod Anda memerlukan akses ke AWS layanan lain, maka Anda perlu membuat titik akhir tambahan. Untuk daftar lengkap titik akhir, lihat [AWS layanan yang terintegrasi dengan AWS PrivateLink](#).

Buat VPC

Anda dapat membuat VPC yang memenuhi persyaratan sebelumnya menggunakan salah satu templat berikut: AWS CloudFormation

- [Template 1](#) - Template ini membuat VPC dengan satu subnet pribadi di Outpost dan satu subnet publik di Wilayah. AWS Subnet pribadi memiliki rute ke internet melalui NAT Gateway yang berada di subnet publik di Wilayah. AWS Template ini dapat digunakan untuk membuat cluster lokal di subnet dengan akses internet jalan keluar.
- [Template 2](#) — Template ini membuat VPC dengan satu subnet pribadi di Outpost dan set minimum VPC Endpoint yang diperlukan untuk membuat cluster lokal di subnet yang tidak memiliki akses internet ingress atau egress (juga disebut sebagai subnet pribadi).

Siapkan kluster Amazon EKS lokal di AWS Outposts untuk pemutusan jaringan

Jika jaringan lokal Anda kehilangan konektivitas dengan AWS Cloud, Anda dapat terus menggunakan kluster Amazon EKS lokal Anda di Outpost. Topik ini mencakup bagaimana Anda dapat mempersiapkan cluster lokal Anda untuk pemutusan jaringan dan pertimbangan terkait.

- Cluster lokal memungkinkan stabilitas dan operasi lanjutan selama pemutusan jaringan sementara yang tidak direncanakan. AWS Outposts tetap merupakan penawaran yang terhubung penuh yang bertindak sebagai perpanjangan AWS Cloud di pusat data Anda. Jika jaringan terputus antara Outpost dan AWS Cloud Anda, kami sarankan untuk mencoba memulihkan koneksi Anda. Untuk instruksi, lihat [daftar periksa pemecahan masalah jaringan rak AWS Outposts](#) di Panduan Pengguna Outposts. AWS Untuk informasi selengkapnya tentang cara memecahkan masalah dengan cluster lokal, lihat [the section called “Memecahkan masalah cluster”](#)
- Outposts memancarkan ConnectedStatus metrik yang dapat Anda gunakan untuk memantau status konektivitas Outpost Anda. Untuk informasi selengkapnya, lihat [Metrik Outposts di Panduan Pengguna Outposts AWS](#).
- Cluster lokal menggunakan IAM sebagai mekanisme autentikasi default menggunakan [AWS Identity and Access Management authenticator](#) untuk Kubernetes. IAM tidak tersedia selama

pemutusan jaringan. Jadi, cluster lokal mendukung mekanisme otentikasi alternatif menggunakan x.509 sertifikat yang dapat Anda gunakan untuk terhubung ke cluster Anda selama pemutusan jaringan. Untuk informasi tentang cara mendapatkan dan menggunakan x.509 sertifikat untuk kluster Anda, lihat [the section called “Mengautentikasi ke cluster lokal Anda selama pemutusan jaringan”](#).

- Jika Anda tidak dapat mengakses Route 53 selama pemutusan jaringan, pertimbangkan untuk menggunakan server DNS lokal di lingkungan lokal Anda. Instance control plane Kubernetes menggunakan alamat IP statis. Anda dapat mengonfigurasi host yang Anda gunakan untuk terhubung ke cluster Anda dengan nama host endpoint dan alamat IP sebagai alternatif untuk menggunakan server DNS lokal. Untuk informasi selengkapnya, lihat [DNS](#) di Panduan AWS Pengguna Outposts.
- Jika Anda mengharapkan peningkatan lalu lintas aplikasi selama pemutusan jaringan, Anda dapat menyediakan kapasitas komputasi cadangan di cluster Anda saat terhubung ke cloud. EC2 Instans Amazon sudah termasuk dalam harga AWS Outposts. Jadi, menjalankan instance cadangan tidak memengaruhi biaya AWS penggunaan Anda.
- Selama pemutusan jaringan untuk mengaktifkan operasi pembuatan, pembaruan, dan skala untuk beban kerja, gambar kontainer aplikasi Anda harus dapat diakses melalui jaringan lokal dan cluster Anda harus memiliki kapasitas yang cukup. Cluster lokal tidak meng-host registri kontainer untuk Anda. Jika Pod sebelumnya telah berjalan pada node tersebut, image kontainer akan di-cache pada node. Jika Anda biasanya menarik gambar kontainer aplikasi Anda dari Amazon ECR di cloud, pertimbangkan untuk menjalankan cache atau registri lokal. Cache atau registri lokal sangat membantu jika Anda memerlukan operasi buat, perbarui, dan skala untuk sumber daya beban kerja selama pemutusan jaringan.
- Cluster lokal menggunakan Amazon EBS sebagai kelas penyimpanan default untuk volume persisten dan driver Amazon EBS CSI untuk mengelola siklus hidup volume persisten Amazon EBS. Selama pemutusan jaringan, Pod yang didukung oleh Amazon EBS tidak dapat dibuat, diperbarui, atau diskalkan. Ini karena operasi ini memerlukan panggilan ke Amazon EBS API di cloud. Jika Anda menerapkan beban kerja stateful pada kluster lokal dan memerlukan operasi pembuatan, pembaruan, atau skala selama pemutusan jaringan, pertimbangkan untuk menggunakan mekanisme penyimpanan alternatif.
- Snapshot Amazon EBS tidak dapat dibuat atau dihapus jika AWS Outposts tidak dapat mengakses AWS wilayah yang relevan APIs (seperti untuk APIs Amazon EBS atau Amazon S3).
- Saat mengintegrasikan ALB (Ingress) dengan AWS Certificate Manager (ACM), sertifikat didorong dan disimpan dalam memori instance Outposts AWS ALB Compute. Penghentian TLS saat ini akan terus beroperasi jika terjadi pemutusan hubungan dari Wilayah. AWS Operasi mutasi dalam

konteks ini akan gagal (seperti definisi ingress baru, operasi API sertifikat berbasis ACM baru, skala komputasi ALB, atau rotasi sertifikat). Untuk informasi selengkapnya, lihat [Memecahkan masalah perpanjangan sertifikat terkelola di Panduan Pengguna AWS Certificate Manager](#).

- Log bidang kontrol Amazon EKS di-cache secara lokal pada instance bidang kontrol Kubernetes selama pemutusan jaringan. Setelah tersambung kembali, log dikirim ke CloudWatch Log di AWS Wilayah induk. Anda dapat menggunakan solusi mitra [Prometheus](#), [Grafana](#), atau Amazon EKS untuk memantau kluster secara lokal menggunakan titik akhir metrik server Kubernetes API atau menggunakan Fluent Bit untuk log.
- Jika Anda menggunakan AWS Load Balancer Controller di Outposts untuk lalu lintas aplikasi, Pod yang ada di depan oleh Load Balancer AWS Controller akan terus menerima lalu lintas selama jaringan terputus. Pod baru yang dibuat selama pemutusan jaringan tidak menerima lalu lintas hingga Outpost tersambung kembali ke Cloud. AWS Pertimbangkan untuk mengatur jumlah replika untuk aplikasi Anda saat terhubung ke AWS Cloud untuk mengakomodasi kebutuhan penskalaan Anda selama pemutusan jaringan.
- [Plugin Amazon VPC CNI untuk Kubernetes default ke mode IP sekunder](#). Ini dikonfigurasi dengan `WARM_ENI_TARGET=1`, yang memungkinkan plugin untuk menjaga “full elastic network interface” dari alamat IP yang tersedia. Pertimbangkan untuk mengubah `WARM_ENI_TARGET`, `WARM_IP_TARGET`, dan `MINIMUM_IP_TARGET` nilai sesuai dengan kebutuhan penskalaan Anda selama keadaan terputus. Untuk informasi selengkapnya, lihat file [readme](#) untuk plugin di GitHub. Untuk daftar jumlah maksimum Pod yang didukung oleh setiap jenis instans, lihat [eni-max-podsfile.txt](#) di GitHub.

Mengautentikasi ke cluster lokal Anda selama pemutusan jaringan

AWS Identity and Access Management (IAM) tidak tersedia selama pemutusan jaringan. Anda tidak dapat mengautentikasi ke kluster lokal menggunakan kredensial IAM saat terputus. Namun, Anda dapat terhubung ke cluster Anda melalui jaringan lokal Anda menggunakan x509 sertifikat saat terputus. Anda perlu mengunduh dan menyimpan X509 sertifikat klien untuk digunakan selama pemutusan sambungan. Dalam topik ini, Anda mempelajari cara membuat dan menggunakan sertifikat untuk mengautentikasi ke kluster Anda saat berada dalam keadaan terputus.

1. Buat permintaan penandatanganan sertifikat.

a. Buat permintaan penandatanganan sertifikat.

```
openssl req -new -newkey rsa:4096 -nodes -days 365 \
-keyout admin.key -out admin.csr -subj "/CN=admin"
```

b. Buat permintaan penandatanganan sertifikat di Kubernetes.

```
BASE64_CSR=$(cat admin.csr | base64 -w 0)
cat << EOF > admin-csr.yaml
apiVersion: certificates.k8s.io/v1
kind: CertificateSigningRequest
metadata:
  name: admin-csr
spec:
  signerName: kubernetes.io/kube-apiserver-client
  request: ${BASE64_CSR}
  usages:
    - client auth
EOF
```

2. Buat permintaan penandatanganan sertifikat menggunakan `kubectl`.

```
kubectl create -f admin-csr.yaml
```

3. Periksa status permintaan penandatanganan sertifikat.

```
kubectl get csr admin-csr
```

Contoh output adalah sebagai berikut.

NAME	AGE	REQUESTOR	CONDITION
admin-csr	11m	kubernetes-admin	Pending

Kubernetes membuat permintaan penandatanganan sertifikat.

4. Menyetujui permintaan penandatanganan sertifikat.

```
kubectl certificate approve admin-csr
```

5. Periksa kembali status permintaan penandatanganan sertifikat untuk persetujuan.

```
kubectl get csr admin-csr
```

Contoh output adalah sebagai berikut.

NAME	AGE	REQUESTOR	CONDITION

```
admin-csr 11m kubernetes-admin Approved
```

6. Ambil dan verifikasi sertifikat.

a. Ambil sertifikat.

```
kubectl get csr admin-csr -o jsonpath='{.status.certificate}' | base64 --decode > admin.crt
```

b. Verifikasi sertifikat.

```
cat admin.crt
```

7. Buat pengikatan peran klaster untuk admin pengguna.

```
kubectl create clusterrolebinding admin --clusterrole=cluster-admin \
  --user=admin --group=system:masters
```

8. Hasilkan kubeconfig dengan cakupan pengguna untuk status terputus.

Anda dapat membuat kubeconfig file menggunakan admin sertifikat yang diunduh. Ganti *my-cluster* dan *apiserver-endpoint* dalam perintah berikut.

```
aws eks describe-cluster --name my-cluster \
  --query "cluster.certificateAuthority" \
  --output text | base64 --decode > ca.crt
```

```
kubectl config --kubeconfig admin.kubeconfig set-cluster my-cluster \
  --certificate-authority=ca.crt --server apiserver-endpoint --embed-certs
```

```
kubectl config --kubeconfig admin.kubeconfig set-credentials admin \
  --client-certificate=admin.crt --client-key=admin.key --embed-certs
```

```
kubectl config --kubeconfig admin.kubeconfig set-context admin@my-cluster \
  --cluster my-cluster --user admin
```

```
kubectl config --kubeconfig admin.kubeconfig use-context admin@my-cluster
```

9. Lihat kubeconfig file Anda.


```
kubectl get nodes --kubeconfig admin.kubeconfig
```

10. Jika Anda memiliki layanan yang sudah diproduksi di Outpost Anda, lewati langkah ini. Jika Amazon EKS adalah satu-satunya layanan yang berjalan di Outpost Anda dan Outpost saat ini tidak dalam produksi, Anda dapat mensimulasikan pemutusan jaringan. Sebelum Anda masuk ke produksi dengan cluster lokal Anda, simulasikan pemutusan untuk memastikan bahwa Anda dapat mengakses kluster Anda saat berada dalam keadaan terputus.

- a. Terapkan aturan firewall pada perangkat jaringan yang menghubungkan Outpost Anda ke AWS Wilayah. Ini memutus tautan layanan dari Outpost. Anda tidak dapat membuat instance baru. Saat ini instans yang sedang berjalan kehilangan konektivitas ke AWS Wilayah dan internet.
- b. Anda dapat menguji koneksi ke cluster lokal Anda saat terputus menggunakan x509 sertifikat. Pastikan untuk mengubah `admin.kubeconfig` ke `admin.kubeconfig` yang Anda buat pada langkah sebelumnya. Ganti `my-cluster` dengan nama cluster lokal Anda.

```
kubectl config use-context admin@my-cluster --kubeconfig admin.kubeconfig
```

Jika Anda melihat ada masalah dengan kluster lokal Anda saat berada dalam keadaan terputus, sebaiknya buka tiket dukungan.

Pilih jenis instans dan grup penempatan untuk kluster Amazon EKS di AWS Outposts berdasarkan pertimbangan kapasitas

Topik ini memberikan panduan untuk memilih jenis instans bidang kontrol Kubernetes dan (opsional) menggunakan grup penempatan untuk memenuhi persyaratan ketersediaan tinggi untuk kluster Amazon EKS lokal Anda di Outpost.

Sebelum Anda memilih tipe instans (seperti `m5`, `c5`, atau `r5`) yang akan digunakan untuk control plane Kubernetes kluster lokal Anda di Outposts, konfirmasi jenis instance yang tersedia pada konfigurasi Outpost Anda. Setelah Anda mengidentifikasi jenis instans yang tersedia, pilih ukuran instans (seperti `large`, `2xlarge`, atau `3xlarge`) berdasarkan jumlah node yang dibutuhkan beban kerja Anda. Tabel berikut memberikan rekomendasi untuk memilih ukuran instance.

Note

Ukuran instans harus ditempatkan di Outposts Anda. Pastikan Anda memiliki kapasitas yang cukup untuk tiga contoh ukuran yang tersedia di Outposts Anda selama masa pakai cluster

lokal Anda. Untuk daftar jenis EC2 instans Amazon yang tersedia, lihat bagian Komputasi dan penyimpanan di fitur rak [AWS Outposts](#).

Jumlah node	Ukuran instance bidang kontrol Kubernetes
1—20	large
21—100	xlarge
101—250	2xlarge
251—500	4xlarge

Penyimpanan untuk bidang kontrol Kubernetes membutuhkan 246 GB penyimpanan Amazon EBS untuk setiap cluster lokal untuk memenuhi IOPS yang diperlukan. etcd Saat kluster lokal dibuat, volume Amazon EBS disediakan secara otomatis untuk Anda.

Kontrol penempatan pesawat

Jika Anda tidak menentukan grup penempatan dengan `OutpostConfig.ControlPlanePlacement.GroupName` properti, EC2 instans Amazon yang disediakan untuk bidang kontrol Kubernetes Anda tidak menerima penegakan penempatan perangkat keras tertentu di seluruh kapasitas dasar yang tersedia di Outpost Anda.

Anda dapat menggunakan grup penempatan untuk memenuhi persyaratan ketersediaan tinggi untuk kluster Amazon EKS lokal Anda di Outpost. Dengan menentukan grup penempatan selama pembuatan kluster, Anda memengaruhi penempatan instance bidang kontrol Kubernetes. Instans tersebar di perangkat keras independen yang mendasari (rak atau host), meminimalkan dampak instance yang berkorelasi pada peristiwa kegagalan perangkat keras.

Jenis spread yang dapat Anda konfigurasi tergantung pada jumlah rak Outpost yang Anda miliki dalam penerapan Anda.

- Deployment dengan satu atau dua rak fisik dalam satu Outpost logis — Anda harus memiliki setidaknya tiga host yang dikonfigurasi dengan tipe instans yang Anda pilih untuk instance control plane Kubernetes Anda. Grup penempatan spread menggunakan spread tingkat host memastikan

bahwa semua instans pesawat kontrol Kubernetes berjalan pada host yang berbeda di dalam rak dasar yang tersedia dalam penyebaran Outpost Anda.

- Deployment dengan tiga atau lebih rak fisik dalam satu Outpost logis — Anda harus memiliki setidaknya tiga host yang dikonfigurasi dengan tipe instans yang Anda pilih untuk instance control plane Kubernetes Anda. Grup penempatan spread menggunakan spread tingkat rak memastikan bahwa semua instance pesawat kontrol Kubernetes berjalan di rak yang berbeda dalam penerapan Outpost Anda. Anda dapat menggunakan grup penempatan spread tingkat host seperti yang dijelaskan pada opsi sebelumnya.

Anda bertanggung jawab untuk membuat grup penempatan yang diinginkan. Anda menentukan grup penempatan saat memanggil `CreateCluster` API. Untuk informasi selengkapnya tentang grup penempatan dan cara membuatnya, lihat [Grup Penempatan](#) di Panduan EC2 Pengguna Amazon.

- Ketika grup penempatan ditentukan, harus ada kapasitas slotted yang tersedia di Outpost Anda agar berhasil membuat kluster Amazon EKS lokal. Kapasitas bervariasi berdasarkan apakah Anda menggunakan tipe host atau rack spread. Jika tidak ada kapasitas yang cukup, cluster tetap dalam `Creating` keadaan. Anda dapat memeriksa `Insufficient Capacity Error` bidang kesehatan respons [DescribeCluster](#) API. Anda harus membebaskan kapasitas untuk proses penciptaan untuk maju.
- Selama pembaruan platform cluster lokal dan versi Amazon EKS, instans control plane Kubernetes dari kluster Anda akan digantikan oleh instance baru menggunakan strategi pembaruan bergulir. Selama proses penggantian ini, setiap instance bidang kontrol dihentikan, membebaskan slotnya masing-masing. Sebuah instance baru yang diperbarui disediakan di tempatnya. Instance yang diperbarui dapat ditempatkan di slot yang dirilis. Jika slot dikonsumsi oleh instance lain yang tidak terkait dan tidak ada lagi kapasitas yang tersisa yang menghormati persyaratan topologi spread yang diperlukan, maka cluster tetap dalam keadaan `Updating`. Anda dapat melihat masing-masing `Insufficient Capacity Error` di bidang kesehatan respons [DescribeCluster](#) API. Anda harus membebaskan kapasitas sehingga proses pembaruan dapat berkembang dan membangun kembali tingkat ketersediaan tinggi sebelumnya.
- Anda dapat membuat maksimal 500 grup penempatan per akun di setiap AWS Wilayah. Untuk informasi selengkapnya, lihat [Aturan dan batasan umum](#) di Panduan EC2 Pengguna Amazon.

Memecahkan masalah cluster Amazon EKS lokal di Outposts AWS

Topik ini mencakup beberapa kesalahan umum yang mungkin Anda lihat saat menggunakan kluster lokal dan cara memecahkan masalah. Cluster lokal mirip dengan cluster Amazon EKS di cloud, tetapi ada beberapa perbedaan dalam cara mereka dikelola oleh Amazon EKS.

Important

Jangan pernah menghentikan instans Kubernetes pesawat kontrol kluster lokal EKS terkelola yang berjalan di Outpost kecuali diinstruksikan secara eksplisit oleh Support. AWS Mengakhiri instance ini menimbulkan risiko terhadap ketersediaan layanan klaster lokal, termasuk hilangnya klaster lokal jika beberapa instance dihentikan secara bersamaan. Instance Kubernetes bidang kontrol kluster lokal EKS diidentifikasi oleh tag `eks-local:controlplane-name` pada konsol instance. EC2

Perilaku API

Cluster lokal dibuat melalui Amazon EKS API, tetapi dijalankan secara asinkron. Ini berarti bahwa permintaan ke Amazon EKS API segera dikembalikan untuk cluster lokal. Namun, permintaan ini mungkin berhasil, gagal cepat karena kesalahan validasi input, atau gagal dan memiliki kesalahan validasi deskriptif. Perilaku ini mirip dengan Kubernetes API.

Cluster lokal tidak bertransisi ke FAILED status. Amazon EKS mencoba untuk mendamaikan status cluster dengan status yang diinginkan pengguna secara berkelanjutan. Akibatnya, klaster lokal mungkin tetap dalam CREATING status untuk jangka waktu yang lama sampai masalah mendasar diselesaikan.

Jelaskan bidang kesehatan cluster

Masalah klaster lokal dapat ditemukan menggunakan perintah AWS CLI Amazon EKS [CLI klaster deskripsikan](#). Masalah cluster lokal muncul oleh `cluster.health` bidang respons `describe-cluster` perintah. Pesan yang terkandung dalam bidang ini mencakup kode kesalahan, pesan deskriptif, dan sumber daya IDs terkait. Informasi ini tersedia melalui Amazon EKS API dan AWS CLI saja. Dalam contoh berikut, ganti *my-cluster* dengan nama cluster lokal Anda.

```
aws eks describe-cluster --name my-cluster --query 'cluster.health'
```

Contoh output adalah sebagai berikut.

```
{
  "issues": [
    {
      "code": "ConfigurationConflict",
      "message": "The instance type 'm5.large' is not supported in Outpost 'my-outpost-arn'.",
      "resourceIds": [
        "my-cluster-arn"
      ]
    }
  ]
}
```

Jika masalahnya tidak dapat diperbaiki, Anda mungkin perlu menghapus cluster lokal dan membuat yang baru. Misalnya, mencoba menyediakan cluster dengan tipe instance yang tidak tersedia di Outpost Anda. Tabel berikut mencakup kesalahan umum terkait kesehatan.

Skenario kesalahan	Kode	Pesan	ResourceIds
Subnet yang disediakan tidak dapat ditemukan.	ResourceNotFound	The subnet ID <i>subnet-id</i> does not exist	Semua subnet yang disediakan IDs
Subnet yang disediakan bukan milik VPC yang sama.	ConfigurationConflict	Subnets specified must belong to the same VPC	Semua subnet yang disediakan IDs
Beberapa subnet yang disediakan bukan milik Outpost yang ditentukan.	ConfigurationConflict	Subnet <i>subnet-id</i> expected to be in <i>outpost-arn</i> , but is in <i>other-outpost-arn</i>	ID subnet bermasalah
Beberapa subnet yang disediakan bukan milik Pos Luar mana pun.	ConfigurationConflict	Subnet <i>subnet-id</i> is not part of any Outpost	ID subnet bermasalah

Skenario kesalahan	Kode	Pesan	ResourceIds
Beberapa subnet yang disediakan tidak memiliki cukup alamat gratis untuk membuat antarmuka jaringan elastis untuk instance bidang kontrol.	ResourceLimitExceeded	The specified subnet does not have enough free addresses to satisfy the request.	ID subnet bermasalah
Jenis instans bidang kontrol yang ditentukan tidak didukung di Outpost Anda.	ConfigurationConflict	The instance type <i>type</i> is not supported in Outpost <i>outpost-arn</i>	ARN klaster
Anda menghentikan EC2 instans Amazon bidang kontrol atau <code>run-instance</code> berhasil, tetapi status yang diamati berubah menjadi <code>Terminated</code> . Ini dapat terjadi untuk jangka waktu tertentu setelah Outpost Anda terhubung kembali dan kesalahan internal Amazon EBS menyebabkan alur kerja EC2 internal Amazon gagal.	InternalFailure	EC2 instance state "Terminated" is unexpected	ARN klaster

Skenario kesalahan	Kode	Pesan	ResourceIds
Anda memiliki kapasitas yang tidak mencukupi di Pos Luar Anda. Hal ini juga dapat terjadi ketika sebuah cluster sedang dibuat jika Outpost terputus dari Region. AWS	ResourceLimitExceeded	There is not enough capacity on the Outpost to launch or start the instance.	ARN klaster
Akun Anda melebihi kuota grup keamanan Anda.	ResourceLimitExceeded	Pesan galat yang dikembalikan oleh Amazon EC2 API	ID VPC Target
Akun Anda melebihi kuota elastic network interface Anda.	ResourceLimitExceeded	Pesan galat yang dikembalikan oleh Amazon EC2 API	ID subnet target
Instance control plane tidak dapat dijangkau melalui Systems Manager. AWS Untuk resolusi, lihat the section called “Instance control plane tidak dapat dijangkau melalui Systems Manager AWS” .	ClusterUnreachable	Instans pesawat kontrol Amazon EKS tidak dapat dijangkau melalui SSM. Harap verifikasi SSM dan konfigurasi jaringan Anda, dan rujuk dokumentasi pemecahan masalah EKS on Outposts.	EC2 Contoh Amazon IDs

Skenario kesalahan	Kode	Pesan	ResourceIds
Terjadi kesalahan saat mendapatkan detail untuk grup keamanan terkelola atau elastic network interface.	Berdasarkan kode kesalahan EC2 klien Amazon.	Pesan galat yang dikembalikan oleh Amazon EC2 API	Semua grup keamanan terkelola IDs
Terjadi kesalahan saat mengotorisasi atau mencabut aturan masuknya grup keamanan. Ini berlaku untuk kelompok keamanan cluster dan pesawat kontrol.	Berdasarkan kode kesalahan EC2 klien Amazon.	Pesan galat yang dikembalikan oleh Amazon EC2 API	ID grup keamanan bermasalah
Terjadi kesalahan saat menghapus elastic network interface untuk instance control plane.	Berdasarkan kode kesalahan EC2 klien Amazon.	Pesan galat yang dikembalikan oleh Amazon EC2 API	ID antarmuka elastis network yang bermasalah

Tabel berikut mencantumkan kesalahan dari AWS layanan lain yang disajikan di bidang kesehatan `describe-cluster` respons.

Kode EC2 kesalahan Amazon	Kode masalah kesehatan cluster	Deskripsi
AuthFailure	AccessDenied	Kesalahan ini dapat terjadi karena berbagai alasan. Alasan paling umum adalah Anda secara tidak sengaja menghapus tag yang digunakan layanan untuk

Kode EC2 kesalahan Amazon	Kode masalah kesehatan cluster	Deskripsi
		menutupi kebijakan peran terkait layanan dari bidang kontrol. Jika ini terjadi, Amazon EKS tidak dapat lagi mengelola dan memantau AWS sumber daya ini.
UnauthorizedOperation	AccessDenied	Kesalahan ini dapat terjadi karena berbagai alasan. Alasan paling umum adalah Anda secara tidak sengaja menghapus tag yang digunakan layanan untuk menutupi kebijakan peran terkait layanan dari bidang kontrol. Jika ini terjadi, Amazon EKS tidak dapat lagi mengelola dan memantau AWS sumber daya ini.
InvalidSubnetID.NotFound	ResourceNotFound	Kesalahan ini terjadi ketika subnet ID untuk aturan masuknya grup keamanan tidak dapat ditemukan.
InvalidPermission.NotFound	ResourceNotFound	Kesalahan ini terjadi ketika izin untuk aturan masuknya grup keamanan tidak benar.
InvalidGroup.NotFound	ResourceNotFound	Kesalahan ini terjadi ketika grup aturan masuk grup keamanan tidak dapat ditemukan.

Kode EC2 kesalahan Amazon	Kode masalah kesehatan cluster	Deskripsi
InvalidNetworkInterfaceID.NotFound	ResourceNotFound	Kesalahan ini terjadi ketika ID antarmuka jaringan untuk aturan masuknya grup keamanan tidak dapat ditemukan.
InsufficientFreeAddressesInSubnet	ResourceLimitExceeded	Kesalahan ini terjadi ketika kuota sumber daya subnet terlampaui.
InsufficientCapacityOnOutpost	ResourceLimitExceeded	Kesalahan ini terjadi ketika kuota kapasitas pos terlampaui.
NetworkInterfaceLimitExceeded	ResourceLimitExceeded	Kesalahan ini terjadi ketika kuota elastic network interface terlampaui.
SecurityGroupLimitExceeded	ResourceLimitExceeded	Kesalahan ini terjadi ketika kuota grup keamanan terlampaui.

Kode EC2 kesalahan Amazon	Kode masalah kesehatan cluster	Deskripsi
VcpuLimitExceeded	ResourceLimitExceeded	Ini diamati saat membuat EC2 instance Amazon di akun baru. Kesalahannya mungkin mirip dengan yang berikut:" You have requested more vCPU capacity than your current vCPU limit of 32 allows for the instance bucket that the specified instance type belongs to. Please visit http://aws.amazon.com/contact-us/ec2-request to request an adjustment to this limit."
InvalidParameterValue	ConfigurationConflict	Amazon EC2 mengembalikan kode kesalahan ini jika jenis instance yang ditentukan tidak didukung di Outpost.
Semua kegagalan lainnya	InternalFailure	Tidak ada

Tidak dapat membuat atau memodifikasi kluster

Cluster lokal memerlukan izin dan kebijakan yang berbeda dari kluster Amazon EKS yang di-host di cloud. Jika kluster gagal membuat dan menghasilkan `InvalidPermissions` kesalahan, periksa kembali apakah peran kluster yang Anda gunakan memiliki kebijakan `EKSLocal OutpostClusterPolicy` terkelola [Amazon](#) yang dilampirkan padanya. Semua panggilan API lainnya memerlukan set izin yang sama dengan kluster Amazon EKS di cloud.

Cluster terjebak dalam **CREATING** keadaan

Jumlah waktu yang dibutuhkan untuk membuat cluster lokal bervariasi tergantung pada beberapa faktor. Faktor-faktor ini termasuk konfigurasi jaringan Anda, konfigurasi Outpost, dan konfigurasi cluster. Secara umum, cluster lokal dibuat dan berubah ACTIVE status dalam 15-20 menit. Jika cluster lokal tetap dalam CREATING status, Anda dapat `describe-cluster` meminta informasi tentang penyebabnya di bidang `cluster.health` output.

Masalah yang paling umum adalah sebagai berikut:

- Cluster Anda tidak dapat terhubung ke instance control plane dari AWS Region tempat Systems Manager berada. Anda dapat memverifikasi ini dengan menelepon `aws ssm start-session --target instance-id` dari host benteng In-region. Jika perintah itu tidak berfungsi, periksa apakah Systems Manager berjalan pada instance control plane. Atau, solusi lain adalah menghapus cluster dan kemudian membuatnya kembali.
- Instans bidang kontrol gagal dibuat karena izin kunci KMS untuk volume EBS. Dengan kunci KMS yang dikelola pengguna untuk volume EBS terenkripsi, instance bidang kontrol akan berakhir jika kunci tidak dapat diakses. Jika instance dihentikan, alihkan ke kunci KMS AWS terkelola atau pastikan bahwa kebijakan kunci terkelola pengguna Anda memberikan izin yang diperlukan ke peran klaster.
- Instans pesawat kontrol Systems Manager mungkin tidak memiliki akses internet. Periksa apakah subnet yang Anda berikan saat membuat cluster memiliki gateway NAT dan VPC dengan gateway internet. Gunakan penganalisis jangkauan VPC untuk memverifikasi bahwa instance bidang kontrol dapat mencapai gateway internet. Untuk informasi selengkapnya, lihat [Memulai dengan VPC Reachability](#) Analyzer.
- Peran ARN yang Anda berikan adalah kebijakan yang hilang. Periksa apakah [kebijakan AWS terkelola: Amazon EKSLocal OutpostClusterPolicy](#) telah dihapus dari peran. Ini juga dapat terjadi jika AWS CloudFormation tumpukan salah dikonfigurasi.
- Semua subnet yang disediakan harus dikaitkan dengan Outpost yang sama dan harus saling menjangkau. Saat beberapa subnet ditentukan saat kluster dibuat, Amazon EKS mencoba menyebarkan instance bidang kontrol di beberapa subnet.
- Grup keamanan terkelola Amazon EKS diterapkan di elastic network interface. Namun, elemen konfigurasi lain seperti aturan firewall NACL mungkin bertentangan dengan aturan untuk elastic network interface.

Konfigurasi DNS VPC dan subnet salah dikonfigurasi atau hilang

Tinjau [Buat VPC dan subnet untuk cluster Amazon EKS di Outposts](#). AWS

Cluster terjebak dalam **UPDATING** keadaan

Amazon EKS secara otomatis memperbarui semua cluster lokal yang ada ke versi platform terbaru untuk versi minor Kubernetes yang sesuai. Untuk informasi lebih lanjut tentang versi platform, silakan lihat [the section called “Versi platform EKS”](#).

Selama peluncuran versi platform otomatis, status klaster berubah menjadi. UPDATING Proses pembaruan terdiri dari penggantian semua instans control-plane Kubernetes dengan yang baru yang berisi jalur keamanan terbaru dan perbaikan bug yang dirilis untuk masing-masing versi minor Kubernetes. Secara umum, proses pembaruan platform cluster lokal selesai dalam waktu kurang dari 30 menit dan klaster berubah kembali ke ACTIVE status. Jika klaster lokal tetap dalam UPDATING status untuk jangka waktu yang lama, Anda dapat menelepon `describe-cluster` untuk memeriksa informasi tentang penyebab di bidang `cluster.health` output.

Amazon EKS memastikan setidaknya 2 dari 3 instans control-plane Kubernetes adalah node cluster yang sehat dan operasional untuk menjaga ketersediaan klaster lokal dan mencegah gangguan layanan. Jika klaster lokal terhenti dalam UPDATING keadaan, biasanya karena ada beberapa masalah infrastruktur atau konfigurasi yang mencegah ketersediaan minimum dua instance dijamin jika proses berlanjut. Jadi proses pembaruan berhenti berkembang untuk melindungi gangguan layanan cluster lokal.

Penting untuk memecahkan masalah klaster lokal yang terjebak dalam UPDATING status dan mengatasi penyebab akar sehingga proses pembaruan dapat menyelesaikan dan memulihkan cluster lokal kembali ACTIVE dengan ketersediaan tinggi 3 instance bidang kontrol Kubernetes.

Jangan menghentikan Kubernetes instans kluster lokal EKS yang dikelola di Outposts kecuali secara eksplisit diinstruksikan oleh Support. AWS Ini sangat penting untuk cluster lokal yang terjebak dalam UPDATING keadaan karena ada kemungkinan besar bahwa node bidang kontrol lain tidak sepenuhnya sehat dan menghentikan instance yang salah dapat menyebabkan gangguan layanan dan risiko kehilangan data cluster lokal.

Masalah yang paling umum adalah sebagai berikut:

- Satu atau beberapa instance bidang kontrol tidak dapat terhubung ke Manajer Sistem karena perubahan konfigurasi jaringan sejak cluster lokal pertama kali dibuat. Anda dapat memverifikasi ini dengan menelepon `aws ssm start-session --target instance-id` dari host benteng In-region. Jika perintah itu tidak berfungsi, periksa apakah Systems Manager berjalan pada instance control plane.

- Instans bidang kontrol baru gagal dibuat karena izin kunci KMS untuk volume EBS. Dengan kunci KMS yang dikelola pengguna untuk volume EBS terenkripsi, instance bidang kontrol akan berakhir jika kunci tidak dapat diakses. Jika instance dihentikan, alihkan ke kunci KMS AWS terkelola atau pastikan bahwa kebijakan kunci terkelola pengguna Anda memberikan izin yang diperlukan ke peran klaster.
- Instans pesawat kontrol Systems Manager mungkin telah kehilangan akses internet. Periksa apakah subnet yang disediakan saat Anda membuat cluster memiliki gateway NAT dan VPC dengan gateway internet. Gunakan penganalisis jangkauan VPC untuk memverifikasi bahwa instance bidang kontrol dapat mencapai gateway internet. Untuk informasi selengkapnya, lihat [Memulai dengan VPC Reachability](#) Analyzer. Jika jaringan pribadi Anda tidak memiliki koneksi internet keluar, pastikan bahwa semua titik akhir VPC dan titik akhir gateway yang diperlukan masih ada di subnet Regional dari cluster Anda (lihat). [the section called “Akses subnet ke layanan AWS”](#)
- Peran ARN yang Anda berikan adalah kebijakan yang hilang. Periksa apakah [kebijakan AWS terkelola: Amazon EKSLocal OutpostClusterPolicy](#) tidak dihapus dari peran.
- Salah satu instance control-plane Kubernetes baru mungkin mengalami kegagalan bootstrap yang tidak terduga. Silakan ajukan tiket ke [AWS Support Center](#) untuk panduan lebih lanjut tentang pemecahan masalah dan pengumpulan log dalam kasus luar biasa ini.

Tidak dapat menggabungkan node ke cluster

- Masalah AMI:
 - Anda menggunakan AMI yang tidak didukung. Anda harus menggunakan [v20220620](#) atau yang lebih baru untuk [Buat node dengan Amazon Linux Amazon AMIs EKS yang dioptimalkan Amazon Linux](#) yang dioptimalkan.
 - Jika Anda menggunakan AWS CloudFormation template untuk membuat node, pastikan itu tidak menggunakan AMI yang tidak didukung.
- Kehilangan AWS IAM Authenticator ConfigMap — Jika hilang, Anda harus membuatnya. Untuk informasi selengkapnya, lihat [the section called “Terapkan aws-authConfigMap ke cluster Anda”](#).
- Kelompok keamanan yang salah digunakan — Pastikan untuk digunakan eks-cluster-sg-**cluster-name-uniqueid** untuk kelompok keamanan node pekerja Anda. Grup keamanan yang dipilih diubah oleh AWS CloudFormation untuk memungkinkan grup keamanan baru setiap kali tumpukan digunakan.
- Mengikuti langkah-langkah VPC tautan pribadi yang tidak terduga — Data CA yang salah (--b64-cluster-ca) atau API Endpoint (--apiserver-endpoint) diteruskan.

Mengumpulkan log

Ketika Outpost terputus dari AWS Region yang terkait dengannya, kluster Kubernetes kemungkinan akan terus bekerja secara normal. Namun, jika kluster tidak berfungsi dengan baik, ikuti langkah-langkah pemecahan masalah di [Siapkan kluster Amazon EKS lokal di AWS Outposts](#) untuk pemutusan jaringan. Jika Anda mengalami masalah lain, hubungi AWS Support. AWS Support dapat memandu Anda dalam mengunduh dan menjalankan alat pengumpulan log. Dengan begitu, Anda dapat mengumpulkan log dari instance control plane cluster Kubernetes dan mengirimkannya ke Support AWS Support untuk penyelidikan lebih lanjut.

Instance control plane tidak dapat dijangkau melalui Systems Manager AWS

Jika instans bidang kontrol Amazon EKS tidak dapat dijangkau melalui AWS Systems Manager (Systems Manager), Amazon EKS menampilkan error berikut untuk kluster Anda.

```
Amazon EKS control plane instances are not reachable through SSM. Please verify your SSM and network configuration, and reference the EKS on Outposts troubleshooting documentation.
```

Untuk mengatasi masalah ini, pastikan VPC dan subnet Anda memenuhi persyaratan di [Buat VPC dan subnet untuk kluster Amazon EKS di AWS Outposts dan](#) Anda menyelesaikan langkah-langkah dalam Menyiapkan [Session Manager di Panduan Pengguna Systems Manager](#). AWS

Buat node Amazon Linux di AWS Outposts

Topik ini menjelaskan bagaimana Anda dapat meluncurkan grup Auto Scaling dari node Amazon Linux di Outpost yang mendaftar dengan kluster Amazon EKS Anda. Cluster dapat berada di AWS Cloud atau di Outpost.

- Pos terdepan yang ada. Untuk informasi selengkapnya, lihat [Apa itu AWS Outposts](#).
- Sebuah kluster Amazon EKS yang sudah ada. Untuk menerapkan cluster di AWS Cloud, lihat [the section called "Membuat kluster"](#). Untuk menyebarkan cluster di Outpost, lihat. [the section called "Jalankan cluster lokal"](#)
- Misalkan Anda membuat node di cluster di AWS Cloud dan Anda memiliki subnet di AWS Wilayah tempat Anda mengaktifkan AWS Outposts, AWS Wavelength, atau Local Zones. AWS Kemudian, subnet tersebut seharusnya tidak diteruskan saat Anda membuat cluster Anda. Jika Anda membuat node di cluster di Outpost, Anda pasti telah melewati subnet Outpost saat membuat cluster Anda.

- (Direkomendasikan untuk cluster di AWS Cloud) Plugin Amazon VPC CNI untuk add-on Kubernetes dikonfigurasi dengan peran IAM sendiri yang memiliki kebijakan IAM yang diperlukan yang melekat padanya. Untuk informasi selengkapnya, lihat [the section called “Konfigurasi untuk IRSA”](#). Cluster lokal tidak mendukung peran IAM untuk akun layanan.

Anda dapat membuat grup node Amazon Linux yang dikelola sendiri dengan `eksctl` atau AWS Management Console (dengan AWS CloudFormation templat). Anda juga dapat menggunakan Terraform.

Anda dapat membuat grup node yang dikelola sendiri untuk kluster lokal dengan alat berikut yang dijelaskan di halaman ini:

- [the section called “eksctl”](#)
- [the section called “AWS Management Console”](#)

Important

- Grup node yang dikelola sendiri menyertakan EC2 instans Amazon di akun Anda. Instans ini tidak ditingkatkan secara otomatis saat Anda atau Amazon EKS memperbarui versi pesawat kontrol atas nama Anda. Grup node yang dikelola sendiri tidak memiliki indikasi apa pun di konsol bahwa ia perlu diperbarui. Anda dapat melihat versi `kubelet` yang diinstal pada sebuah simpul dengan memilih simpul di dalam daftar Simpul pada tab Gambaran Umum kluster Anda untuk menentukan simpul-simpul yang perlu diperbarui. Anda harus memperbarui simpul secara manual. Untuk informasi selengkapnya, lihat [the section called “Metode pembaruan”](#).
- Sertifikat yang digunakan oleh `kubelet` pada node yang dikelola sendiri diterbitkan dengan kedaluwarsa satu tahun. Secara default, rotasi sertifikat tidak diaktifkan (lihat: <https://kubernetes.io/docs/reference/config-api/kubelet-config.v1beta1/# kubelet-config-k8s-io-v1beta1- KubeletConfiguration>), ini berarti jika Anda memiliki node yang dikelola sendiri yang berjalan selama lebih dari satu tahun, itu tidak akan lagi dapat mengautentikasi ke Kubernetes API.
- Sebagai praktik terbaik, kami menyarankan pelanggan untuk secara teratur memperbarui grup node yang dikelola sendiri untuk menerima CVEs dan patch keamanan dari AMI Amazon EKS terbaru yang dioptimalkan. Memperbarui AMI yang digunakan dalam grup node yang dikelola sendiri juga memicu pembuatan ulang node dan memastikan mereka tidak mengalami masalah karena sertifikat `kubelet` yang kedaluwarsa.

- Atau Anda juga dapat mengaktifkan rotasi sertifikat klien (lihat: <https://kubernetes.io/docs/tasks/tls/certificate-rotation/>) saat membuat grup node yang dikelola sendiri untuk memastikan sertifikat kubelet diperbarui saat sertifikat saat ini mendekati kedaluwarsa.

eksctl

Untuk meluncurkan node Linux yang dikelola sendiri menggunakan **eksctl**

1. Instal versi 0.210.0 atau yang lebih baru dari alat baris perintah yang diinstal pada perangkat Anda atau AWS CloudShell. Untuk menginstal atau memperbarui **eksctl**, lihat [Instalasi](#) dalam **eksctl** dokumentasi.
2. Jika klaster Anda ada di AWS Cloud dan kebijakan IAM terkelola Amazon eks_CNI_Policy dilampirkan ke peran IAM [node Amazon EKS Anda](#), kami sarankan untuk [menetapkannya ke peran IAM yang Anda](#) kaitkan ke akun layanan Kubernetes sebagai gantinya. `aws-node` Untuk informasi selengkapnya, lihat [the section called “Konfigurasi untuk IRSA”](#). Jika klaster Anda berada di Outpost Anda, kebijakan harus dilampirkan ke peran node Anda.
3. Perintah berikutnya membuat grup simpul dalam klaster yang ada. Cluster harus dibuat menggunakan **eksctl**. Ganti *al-nodes* dengan nama untuk grup node Anda. Nama grup node tidak boleh lebih dari 63 karakter. Itu harus dimulai dengan huruf atau digit, tetapi juga dapat menyertakan tanda hubung dan garis bawah untuk karakter yang tersisa. Ganti *my-cluster* dengan nama klaster Anda. Nama hanya dapat berisi karakter alfanumerik (peka huruf besar/kecil) dan tanda hubung. Itu harus dimulai dengan karakter alfanumerik dan tidak boleh lebih dari 100 karakter. Nama harus unik di dalam AWS Wilayah dan AWS akun tempat Anda membuat klaster. Jika cluster Anda ada di Outpost, ganti *id* dengan ID subnet Outpost. Jika klaster Anda ada di AWS Cloud, ganti *id* dengan ID subnet yang tidak Anda tentukan saat membuat klaster. Ganti *instance-type* dengan jenis instance yang didukung oleh Outpost Anda. Ganti sisanya *example values* dengan nilai Anda sendiri. Simpul dibuat dengan versi Kubernetes yang sama dengan bidang kendali, secara default.

Ganti *instance-type* dengan jenis instance yang tersedia di Outpost Anda.

Ganti *my-key* dengan nama Amazon EC2 key pair atau public key Anda. Kunci ini digunakan untuk SSH ke simpul Anda setelah diluncurkan. Jika Anda belum memiliki EC2 key pair Amazon, Anda dapat membuatnya di AWS Management Console. Untuk informasi selengkapnya, lihat [pasangan EC2 kunci Amazon](#) di Panduan EC2 Pengguna Amazon.

Buat grup simpul Anda dengan perintah berikut.

```
eksctl create nodegroup --cluster my-cluster --name al-nodes --node-type instance-  
type \  
    --nodes 3 --nodes-min 1 --nodes-max 4 --managed=false --node-volume-type gp2 --  
subnet-ids subnet-id
```

Jika klaster Anda di-deploy di AWS Cloud:

- Grup node yang Anda deploy dapat menetapkan IPv4 alamat ke Pod dari blok CIDR yang berbeda dari instance. Untuk informasi selengkapnya, lihat [the section called “Jaringan kustom”](#).
- Grup node yang Anda terapkan tidak memerlukan akses internet keluar. Untuk informasi selengkapnya, lihat [the section called “klaster privat”](#).

Untuk daftar lengkap semua opsi dan default yang tersedia, lihat [AWS Outposts Support](#) dalam dokumentasi. `eksctl`

- Jika node gagal bergabung dengan cluster, lihat [the section called “Simpul gagal untuk bergabung dengan klaster”](#) di [Memecahkan masalah dengan kluster dan node Amazon EKS dan di Troubleshoot cluster Amazon EKS lokal the section called “Tidak dapat menggabungkan node ke cluster”](#) di Outposts. AWS
- Contoh output adalah sebagai berikut. Beberapa baris adalah output sementara node dibuat. Salah satu baris terakhir dari output adalah baris contoh berikutnya.

```
[#] created 1 nodegroup(s) in cluster "my-cluster"
```

4. (Optional) Deploy [aplikasi sampel](#) untuk menguji simpul klaster dan Linux Anda.

AWS Management Console

Langkah 1: Luncurkan node Linux yang dikelola sendiri menggunakan AWS Management Console

1. Unduh versi terbaru dari AWS CloudFormation template.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2022-12-23/  
amazon-eks-nodegroup.yaml
```

2. Buka [konsol AWS CloudFormation](#) .
3. Pilih Buat tumpukan dan kemudian pilih Dengan sumber daya baru (standar).

4. Untuk Menentukan templat, pilih Unggah sebuah file templat dan kemudian pilih Pilih file. Pilih `amazon-eks-nodegroup.yaml` file yang Anda unduh pada langkah sebelumnya dan kemudian pilih Berikutnya.
5. Pada halaman Tentukan detail tumpukan, masukkan parameter berikut yang sesuai, lalu pilih Berikutnya:
 - Nama tumpukan: Pilih nama tumpukan untuk AWS CloudFormation tumpukan Anda. Misalnya, Anda bisa menyebutnya `l-nodes`. Nama hanya dapat berisi karakter alfanumerik (peka huruf besar/kecil) dan tanda hubung. Itu harus dimulai dengan karakter alfanumerik dan tidak boleh lebih dari 100 karakter. Nama harus unik di dalam AWS Wilayah dan AWS akun tempat Anda membuat klaster.
 - ClusterName: Masukkan nama cluster Anda. Jika nama ini tidak cocok dengan nama cluster Anda, node Anda tidak dapat bergabung dengan cluster.
 - ClusterControlPlaneSecurityGroup: Pilih SecurityGroup nilai dari AWS CloudFormation output yang Anda hasilkan saat Anda membuat [VPC](#) Anda.

Langkah-langkah berikut menunjukkan satu operasi untuk mengambil grup yang berlaku.

- i. Buka [konsol Amazon EKS](#).
 - ii. Pilih nama cluster.
 - iii. Pilih tab Jaringan.
 - iv. Gunakan nilai grup keamanan tambahan sebagai referensi saat memilih dari daftar ClusterControlPlaneSecurityGroup tarik-turun.
- NodeGroupName: Masukkan nama untuk grup node Anda. Nama ini dapat digunakan nanti untuk mengidentifikasi grup node Auto Scaling yang dibuat untuk node Anda.
 - NodeAutoScalingGroupMinSize: Masukkan jumlah minimum node yang dapat diskalakan oleh grup Auto Scaling node Anda.
 - NodeAutoScalingGroupDesiredCapacity: Masukkan jumlah node yang diinginkan untuk diskalakan saat tumpukan Anda dibuat.
 - NodeAutoScalingGroupMaxSize: Masukkan jumlah maksimum node yang dapat diskalakan oleh grup Auto Scaling node Anda.
 - NodeInstanceType: Pilih jenis instance untuk node Anda. Jika klaster Anda berjalan di AWS Cloud, maka untuk informasi selengkapnya, lihat [the section called “Jenis EC2 instans Amazon”](#). Jika cluster Anda berjalan di Outpost, maka Anda hanya dapat memilih jenis instans yang tersedia di Outpost Anda.

- `NodeImageIdSSMParam`: Diisi sebelumnya dengan parameter Amazon EC2 Systems Manager dari Amazon EKS baru-baru ini yang dioptimalkan AMI untuk versi Kubernetes variabel. [Untuk menggunakan versi minor Kubernetes berbeda yang didukung dengan Amazon EKS, ganti 1.XX dengan versi lain yang didukung](#). Sebaiknya tentukan versi Kubernetes yang sama dengan kluster Anda.

Untuk menggunakan AMI akselerasi Amazon EKS yang dioptimalkan, ganti `amazon-linux-2` dengan `amazon-linux-2-gpu`. Untuk menggunakan AMI Arm Amazon EKS yang dioptimalkan, ganti `amazon-linux-2` dengan `amazon-linux-2-arm64`.

Note

Node Amazon EKS AMIs didasarkan pada Amazon Linux. Anda dapat melacak peristiwa keamanan atau privasi untuk Amazon Linux di [pusat keamanan Amazon Linux](#) dengan memilih tab untuk versi yang Anda inginkan. Anda juga dapat berlangganan umpan RSS yang berlaku. Kejadian keamanan dan privasi mencakup gambaran umum mengenai masalah, paket apa yang terpengaruh, dan cara memperbarui instans Anda untuk memperbaiki masalah tersebut.

- `NodeImageId`: (Opsional) Jika Anda menggunakan AMI kustom Anda sendiri (bukan AMI yang dioptimalkan Amazon EKS), masukkan ID AMI node untuk AWS Wilayah Anda. Jika Anda menentukan nilai di sini, itu akan mengganti nilai apa pun di `NodeImageIdSSMParam`.
- `NodeVolumeSize`: Tentukan ukuran volume root untuk node Anda, di GiB.
- `NodeVolumeType`: Tentukan jenis volume root untuk node Anda.
- `KeyName`: Masukkan nama key pair Amazon EC2 SSH yang dapat Anda gunakan untuk terhubung menggunakan SSH ke node Anda setelah diluncurkan. Jika Anda belum memiliki EC2 key pair Amazon, Anda dapat membuatnya di AWS Management Console. Untuk informasi selengkapnya, lihat [pasangan EC2 kunci Amazon](#) di Panduan EC2 Pengguna Amazon.

Note

Jika Anda tidak menyediakan key pair di sini, pembuatan AWS CloudFormation stack gagal.

- `BootstrapArguments`: Ada beberapa argumen opsional yang dapat Anda berikan ke node Anda. Untuk informasi lebih lanjut, lihat [informasi penggunaan skrip bootstrap](#) di GitHub. Jika Anda menambahkan node ke Amazon EKS Local Cluster di AWS Outposts (di mana instance control

plane Kubernetes berjalan di AWS Outposts) dan cluster tidak memiliki koneksi internet ingress dan egress (juga dikenal sebagai cluster pribadi), maka Anda harus memberikan argumen bootstrap berikut (sebagai satu baris).

```
--b64-cluster-ca ${CLUSTER_CA} --apiserver-endpoint https://${APISERVER_ENDPOINT}  
--enable-local-outpost true --cluster-id ${CLUSTER_ID}
```

Untuk mengambil nilai untuk `CLUSTER_CA`, `APISERVER_ENDPOINT`, dan `CLUSTER_ID` kluster lokal Amazon EKS Anda, jalankan perintah AWS CLI berikut. Ganti nama cluster dengan nama kluster dan wilayah Anda (misalnya, `us-east-1`) dengan Region kluster Anda. AWS

```
echo "CLUSTER_CA=$(aws eks describe-cluster --name cluster-name --region region --  
query cluster.certificateAuthority.data --output text)"  
  
echo "APISERVER_ENDPOINT=$(aws eks describe-cluster --name cluster-name --region  
region --query cluster.endpoint --output text)"  
  
echo "CLUSTER_ID=$(aws eks describe-cluster --name cluster-name --region region --  
query cluster.id --output text)"
```

- Nonaktifkan IMDSv1: Secara default, setiap node mendukung Layanan Metadata Instans Versi 1 (IMDSv1) dan IMDSv2. Anda dapat menonaktifkan IMDSv1. Untuk mencegah penggunaan node dan Pod future dalam grup node IMDSv1, setel `Disable IMDSv1` ke `true`. Untuk informasi selengkapnya tentang IMDS, lihat [Mengonfigurasi layanan metadata instans](#). Untuk informasi selengkapnya tentang membatasi akses ke node Anda, lihat [Membatasi akses ke profil instance yang ditetapkan ke node pekerja](#).
 - VpcId: Masukkan ID untuk [VPC](#) yang Anda buat. Sebelum memilih VPC, tinjau persyaratan dan pertimbangan [VPC](#).
 - Subnet: Jika cluster Anda berada di Outpost, maka pilih setidaknya satu subnet pribadi di VPC Anda. Sebelum memilih subnet, tinjau [persyaratan dan pertimbangan Subnet](#). Anda dapat melihat subnet mana yang bersifat pribadi dengan membuka setiap subnet link dari tab Networking cluster Anda.
6. Pilih pilihan yang Anda inginkan di halaman Configure stack options, lalu pilih Next.
 7. Pilih kotak centang di sebelah kiri Saya mengakui yang AWS CloudFormation mungkin membuat sumber daya IAM. , dan kemudian pilih Buat tumpukan.
 8. Setelah tumpukan Anda selesai dibuat, pilih tumpukan di konsol dan pilih Outputs.

9. Rekam `NodeInstanceRole` untuk grup node yang telah dibuat. Anda memerlukan ini saat mengonfigurasi simpul Amazon EKS Anda.

Langkah 2: Aktifkan node untuk bergabung dengan cluster Anda

1. Periksa untuk melihat apakah Anda sudah memiliki `aws-authConfigMap`.

```
kubectl describe configmap -n kube-system aws-auth
```

2. Jika Anda ditampilkan `aws-authConfigMap`, maka perbarui sesuai kebutuhan.

- a. Buka `ConfigMap` untuk mengedit.

```
kubectl edit -n kube-system configmap/aws-auth
```

- b. Tambahkan `mapRoles` entri baru sesuai kebutuhan. Tetapkan `roleARN` nilai ke `NodeInstanceRole` nilai yang Anda rekam dalam prosedur sebelumnya.

```
[...]
data:
  mapRoles: |
    - roleARN: <ARN of instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
[...]
```

- c. Simpan file dan keluar dari editor teks Anda.

3. Jika Anda menerima kesalahan yang menyatakan "Error from server (NotFound): configmaps "aws-auth" not found", maka terapkan `aws-authConfigMap`.

- a. Unduh peta konfigurasi.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/
aws-auth-cm.yaml
```

- b. Dalam `aws-auth-cm.yaml` file, atur `roleARN` ke `NodeInstanceRole` nilai yang Anda rekam dalam prosedur sebelumnya. Anda dapat melakukan ini dengan editor teks, atau dengan mengganti `my-node-instance-role` dan menjalankan perintah berikut:

```
sed -i.bak -e 's|<ARN of instance role (not instance profile)>|my-node-instance-role|' aws-auth-cm.yaml
```

- c. Terapkan konfigurasi. Perintah ini mungkin memerlukan waktu beberapa menit untuk diselesaikan.

```
kubectl apply -f aws-auth-cm.yaml
```

4. Perhatikan status simpul Anda dan tunggu sampai simpul mencapai Status Ready.

```
kubectl get nodes --watch
```

Masukkan `Ctrl+C` untuk kembali ke prompt shell.

Note

Jika Anda menerima kesalahan otorisasi atau jenis sumber daya, lihat [the section called “Tidak sah atau akses ditolak \(kubectl\)”](#) di topik pemecahan masalah.

Jika node gagal bergabung dengan cluster, lihat [the section called “Simpul gagal untuk bergabung dengan kluster”](#) di [Memecahkan masalah dengan kluster dan node Amazon EKS](#) dan di [Troubleshoot cluster Amazon EKS lokal](#) the section called [“Tidak dapat menggabungkan node ke cluster”](#) di Outposts. AWS

5. Instal driver Amazon EBS CSI. Untuk informasi selengkapnya, lihat [Instalasi](#) di GitHub. Di bagian Siapkan izin driver, pastikan untuk mengikuti instruksi untuk opsi Menggunakan profil instans IAM. Anda harus menggunakan kelas gp2 penyimpanan. Kelas gp3 penyimpanan tidak didukung.

Untuk membuat kelas gp2 penyimpanan di cluster Anda, selesaikan langkah-langkah berikut.

- a. Jalankan perintah berikut untuk membuat `gp2-storage-class.yaml` file.

```
cat >gp2-storage-class.yaml <<EOF
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
  name: ebs-sc
```

```
provisioner: ebs.csi.aws.com
volumeBindingMode: WaitForFirstConsumer
parameters:
  type: gp2
  encrypted: "true"
allowVolumeExpansion: true
EOF
```

b. Menerapkan manifes ke klaster Anda.

```
kubectl apply -f gp2-storage-class.yaml
```

6. (Hanya node GPU) Jika Anda memilih jenis instans GPU dan AMI akselerasi Amazon EKS yang dioptimalkan, Anda harus menerapkan [plugin perangkat NVIDIA untuk Kubernetes](#) sebagai a di cluster Anda. DaemonSet Ganti **vX.X.X** dengan s-device-plugin versi [NVIDIA/K8](#) yang Anda inginkan sebelum menjalankan perintah berikut.

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/vX.X.X/
deployments/static/nvidia-device-plugin.yml
```

Step3: Tindakan tambahan

1. (Opsional) Deploy [aplikasi sampel](#) untuk menguji simpul klaster dan Linux Anda.
2. Jika cluster Anda digunakan di Outpost, lewati langkah ini. Jika klaster Anda di-deploy di AWS Cloud, informasi berikut bersifat opsional. Jika kebijakan IAM terkelola Amazoneks_CNI_Policy dilampirkan ke peran IAM [node Amazon EKS Anda](#), [sebaiknya tetapkan ke peran IAM yang Anda kaitkan ke akun layanan Kubernetes sebagai gantinya](#). `aws-node` Lihat informasi yang lebih lengkap di [the section called “Konfigurasi untuk IRSA”](#).

Ikhtisar Machine Learning di Amazon EKS

Amazon Elastic Kubernetes Service (EKS) adalah platform Kubernetes terkelola yang memberdayakan organisasi untuk menerapkan, mengelola, dan menskalakan beban kerja AI dan machine learning (ML) dengan fleksibilitas dan kontrol yang tak tertandingi. Dibangun di atas ekosistem Kubernetes open source, EKS memungkinkan Anda memanfaatkan keahlian Kubernetes yang ada, sambil berintegrasi secara mulus dengan alat dan layanan open source. AWS

Baik Anda melatih model skala besar, menjalankan inferensi online waktu nyata, atau menerapkan aplikasi AI generatif, EKS memberikan kinerja, skalabilitas, dan efisiensi biaya yang diminta proyek AI/ML Anda.

Mengapa Memilih EKS untuk AI/ML?

EKS adalah platform Kubernetes terkelola yang membantu Anda menerapkan dan mengelola beban kerja AI/ML yang kompleks. Dibangun di atas ekosistem Kubernetes open source, ia terintegrasi dengan AWS layanan, menyediakan kontrol dan skalabilitas yang diperlukan untuk proyek-proyek lanjutan. Untuk tim yang baru mengenal penerapan AI/ML, skill Kubernetes yang ada ditransfer secara langsung, memungkinkan orkestrasi yang efisien dari beberapa beban kerja.

EKS mendukung segala hal mulai dari kustomisasi sistem operasi hingga penskalaan komputasi, dan fondasi open source-nya mempromosikan fleksibilitas teknologi, melestarikan pilihan untuk keputusan infrastruktur masa depan. Platform ini menyediakan opsi kinerja dan penyetelan yang dibutuhkan beban kerja AI/ML, fitur-fitur pendukung seperti:

- Kontrol klaster penuh untuk menyempurnakan biaya dan konfigurasi tanpa abstraksi tersembunyi
- Latensi sub-detik untuk beban kerja inferensi waktu nyata dalam produksi
- Kustomisasi lanjutan seperti multi-instance GPUs, strategi multi-cloud, dan penyetelan tingkat OS
- Kemampuan untuk memusatkan beban kerja menggunakan EKS sebagai orkestrator terpadu di seluruh jaringan pipa AI/ML

Kasus penggunaan kunci

Amazon EKS menyediakan platform yang kuat untuk berbagai beban kerja AI/ML, mendukung berbagai teknologi dan pola penerapan:

- Inferensi real-time (online): [EKS mendukung prediksi langsung pada data yang masuk, seperti deteksi penipuan, dengan latensi sub-detik menggunakan alat seperti, Triton Inference Server TorchServe, dan pada instans Amazon Inf1 dan Inf2. KServe EC2](#) Beban kerja ini mendapat manfaat dari penskalaan dinamis dengan [Karpenter](#) dan [KEDA](#), sekaligus memanfaatkan Amazon [EFS](#) untuk sharding model di seluruh pod. [Amazon ECR Pull Through Cache \(PTC\)](#) mempercepat pembaruan model, dan volume data [Bottlerocket](#) dengan volume yang dioptimalkan [Amazon](#) EBS memastikan akses data yang cepat.
- Pelatihan model umum: [Organizations memanfaatkan EKS untuk melatih model kompleks pada kumpulan data besar selama periode yang lama menggunakan Kubeflow Training Operator \(KRO\), Ray Serve, dan Torch Distributed Elastic pada instans Amazon EC2 P4d dan Amazon Trn1. EC2](#) Beban kerja ini didukung oleh penjadwalan batch dengan alat seperti Volcano, Yunikorn, dan [Kueue](#). [Amazon EFS](#) memungkinkan berbagi pos pemeriksaan model, dan [Amazon](#) S3 menangani impor/ekspor model dengan kebijakan siklus hidup untuk manajemen versi.
- Retrieval augmented generation (RAG) pipelines: EKS mengelola chatbot dukungan pelanggan dan aplikasi serupa dengan mengintegrasikan proses pengambilan dan pembuatan. [Beban kerja ini sering menggunakan alat seperti Argo Workflows dan Kubeflow untuk orkestrasi, database vektor seperti Pinecone, Weaviate, atau Amazon, dan OpenSearch mengekspos aplikasi ke pengguna melalui Application Load Balancer Controller \(LBC\). NVIDIA NIM mengoptimalkan pemanfaatan GPU, sementara Prometheus dan Grafana memantau penggunaan sumber daya.](#)
- Penerapan model AI generatif: [Perusahaan menerapkan layanan pembuatan konten real-time di EKS, seperti pembuatan teks atau gambar, menggunakan Ray Serve, VLLM, dan Triton Inference Server di Amazon G5 dan akselerator Inferentia. EC2](#) Penerapan ini mengoptimalkan kinerja dan pemanfaatan memori untuk model skala besar. [JupyterHub](#) memungkinkan pengembangan berulang, [Gradio](#) menyediakan antarmuka web sederhana, dan [S3 Mountpoint CSI Driver](#) memungkinkan pemasangan bucket S3 sebagai sistem file untuk mengakses file model besar.
- Inferensi Batch (offline): [Organizations memproses kumpulan data besar secara efisien melalui pekerjaan terjadwal dengan AWS Batch atau Volcano.](#) Beban kerja ini sering menggunakan instans [Inf1](#) dan [Inf2](#) untuk chip AWS [Inferentia](#), EC2 instans EC2 [Amazon](#) G4dn untuk NVIDIA GPUs T4, [atau](#) instans CPU c5 [dan](#) c6i, memaksimalkan pemanfaatan sumber daya selama jam-jam sibuk untuk tugas analitik. Driver [AWS Neuron SDK](#) dan NVIDIA GPU mengoptimalkan kinerja, sementara MIG/TS memungkinkan berbagi GPU. Solusi penyimpanan termasuk [Amazon S3](#) dan Amazon [EFS](#) dan [FSx untuk Lustre](#), dengan driver CSI untuk berbagai kelas penyimpanan. [Manajemen model memanfaatkan alat seperti Pipelines Kubeflow, Argo Workflows, dan Ray Cluster, sementara pemantauan ditangani oleh Prometheus, Grafana, dan alat pemantauan model kustom.](#)

Studi kasus

Pelanggan memilih Amazon EKS karena berbagai alasan, seperti mengoptimalkan penggunaan GPU atau menjalankan beban kerja inferensi real-time dengan latensi sub-detik, seperti yang ditunjukkan dalam studi kasus berikut. Untuk daftar semua studi kasus untuk Amazon EKS, lihat [Kisah Sukses AWS Pelanggan](#).

- [Unitary](#) memproses 26 juta video setiap hari menggunakan AI untuk moderasi konten, membutuhkan throughput tinggi, inferensi latensi rendah dan telah mencapai pengurangan 80% dalam waktu boot kontainer, memastikan respons cepat terhadap peristiwa penskalaan saat lalu lintas berfluktuasi.
- [Miro](#), platform kolaborasi visual yang mendukung 70 juta pengguna di seluruh dunia, melaporkan pengurangan 80% dalam biaya komputasi dibandingkan dengan cluster Kubernetes yang dikelola sendiri sebelumnya.
- [Synthesia](#), yang menawarkan pembuatan video AI generatif sebagai layanan bagi pelanggan untuk membuat video realistis dari petunjuk teks, mencapai peningkatan 30x dalam throughput pelatihan model ML.
- [Harri, yang menyediakan teknologi SDM untuk industri perhotelan, mencapai penskalaan 90% lebih cepat dalam menanggapi lonjakan permintaan dan mengurangi biaya komputasi sebesar 30% dengan bermigrasi ke prosesor Graviton.AWS](#)
- [Ada Support](#), sebuah perusahaan otomatisasi layanan pelanggan bertenaga AI, mencapai pengurangan 15% dalam biaya komputasi di samping peningkatan 30% dalam efisiensi komputasi.
- [Snorkel AI](#), yang melengkapi perusahaan untuk membangun dan mengadaptasi model pondasi dan model bahasa besar, mencapai penghematan biaya lebih dari 40% dengan menerapkan mekanisme penskalaan cerdas untuk sumber daya GPU mereka.

Mulai menggunakan Machine Learning di EKS

Untuk mulai merencanakan dan menggunakan platform Machine Learning dan beban kerja di EKS di AWS cloud, lanjutkan ke [the section called “Memulai dengan ML”](#) bagian tersebut.

Memulai dengan ML

Untuk terjun ke Machine Learning di EKS, mulailah dengan memilih dari pola preskriptif ini untuk dengan cepat mendapatkan cluster EKS dan perangkat lunak dan perangkat lunak dan perangkat keras yang siap untuk mulai menjalankan beban kerja ML. Sebagian besar pola ini didasarkan pada

cetak biru Terraform yang tersedia dari situs Data [di](#) Amazon EKS. Sebelum Anda mulai, berikut adalah beberapa hal yang perlu diingat:

- GPUs atau contoh Neuron diperlukan untuk menjalankan prosedur ini. Kurangnya ketersediaan sumber daya ini dapat menyebabkan prosedur ini gagal selama pembuatan cluster atau penskalaan otomatis node.
- Neuron SDK (contoh berbasis Trainium dan Inferentia) dapat menghemat uang dan lebih tersedia daripada NVIDIA. GPUs Jadi, ketika beban kerja Anda mengizinkannya, kami sarankan Anda mempertimbangkan untuk menggunakan Neuron untuk beban kerja Machine Learning Anda (lihat [Selamat Datang di AWS Neuron](#)).
- Beberapa pengalaman memulai di sini mengharuskan Anda mendapatkan data melalui akun [Hugging](#) Face Anda sendiri.

Untuk memulai, pilih dari pilihan pola berikut yang dirancang untuk membantu Anda mulai menyiapkan infrastruktur untuk menjalankan beban kerja Machine Learning Anda:

- [JupyterHub di EKS](#): Jelajahi [JupyterHub cetak biru](#), yang menampilkan fitur Time Slicing dan MIG, serta konfigurasi multi-penyewa dengan profil. Ini sangat ideal untuk menyebarkan JupyterHub platform skala besar di EKS.
- [Model Bahasa Besar pada AWS Neuron dan RayServe](#): Gunakan [AWS Neuron](#) untuk menjalankan model bahasa besar (LLMs) di Amazon EKS dan akselerator AWS Trainium dan AWS Inferentia. Lihat [Melayani LLMs dengan RayServe dan VIIM di AWS Neuron](#) untuk instruksi tentang menyiapkan platform untuk membuat permintaan inferensi, dengan komponen yang meliputi:
 - AWS Toolkit Neuron SDK untuk pembelajaran mendalam
 - AWS Akselerator inferensia dan Trainium
 - [VIIM - model bahasa panjang variabel \(lihat situs dokumentasi VIIM\)](#)
 - RayServe pustaka penyajian model yang dapat diskalakan (lihat situs [Ray Serve: Scalable and Programmable](#) Serving)
 - Model bahasa Llama-3, menggunakan akun Hugging [Face](#) Anda sendiri.
 - Observabilitas dengan AWS CloudWatch dan Neuron Monitor
 - Buka WebUI
- [Model Bahasa Besar di NVIDIA dan Triton](#): Terapkan beberapa model bahasa besar () LLMs di Amazon EKS dan NVIDIA. GPUs Lihat [Menerapkan Beberapa Model Bahasa Besar dengan NVIDIA Triton Server dan VLLm](#) untuk petunjuk menyiapkan platform untuk membuat permintaan inferensi, dengan komponen yang mencakup:

- NVIDIA Triton Inference Server (lihat situs [Triton](#) Inference Server) GitHub
- [VLLM - model bahasa panjang variabel](#) (lihat situs dokumentasi VLLM)
- Dua model bahasa: mistralai/Mistral-7B-Instruct-v0.2 and meta-llama/Llama -2-7b-chat-hf, menggunakan akun [Hugging Face](#) Anda sendiri.

Melanjutkan dengan ML di EKS

Seiring dengan memilih dari cetak biru yang dijelaskan di halaman ini, ada cara lain Anda dapat melanjutkan melalui dokumentasi ML pada EKS jika Anda mau. Sebagai contoh, Anda dapat:

- Coba tutorial untuk ML di EKS — Jalankan end-to-end tutorial lain untuk membangun dan menjalankan model Machine Learning Anda sendiri di EKS. Lihat [the section called “Coba tutorial untuk ML di EKS”](#).

Untuk meningkatkan pekerjaan Anda dengan ML di EKS, lihat yang berikut ini:

- Mempersiapkan untuk ML — Pelajari cara mempersiapkan ML di EKS dengan fitur seperti reservasi kustom AMIs dan GPU. Lihat [the section called “Siapkan untuk ML”](#).

Bersiaplah untuk cluster ML

Ada beberapa cara untuk meningkatkan pengalaman Machine Learning on EKS Anda. Halaman berikut di bagian ini akan membantu Anda:

- Pahami pilihan Anda untuk menggunakan ML di EKS dan
- Membantu dalam persiapan lingkungan EKS dan ML Anda.

Secara khusus, ini akan membantu Anda:

- Pilih AMIs: AWS menawarkan beberapa yang disesuaikan AMIs untuk menjalankan beban kerja ML di EKS. Lihat [the section called “Jalankan Linux GPU AMIs”](#) dan [the section called “Jalankan Windows GPU AMIs”](#).
- Kustomisasi AMIs: Anda dapat memodifikasi AWS kustom lebih lanjut AMIs untuk menambahkan perangkat lunak dan driver lain yang diperlukan untuk kasus penggunaan khusus Anda. Lihat [the section called “Cadangan GPUs untuk SMN”](#).

- Cadangan GPUs: Karena permintaan GPUs, untuk memastikan bahwa kebutuhan GPUs Anda tersedia saat Anda membutuhkannya, Anda dapat memesan yang GPUs Anda butuhkan terlebih dahulu. Lihat [the section called “Node GPU noda”](#).
- Tambahkan EFA: Tambahkan Adaptor Kain Elastis untuk meningkatkan kinerja jaringan untuk komunikasi cluster antar-node. Lihat [the section called “Cluster pelatihan dengan EFA”](#).
- Gunakan AWSInferentia beban kerja: Buat klaster EKS dengan instans Amazon EC2 Inf1. Lihat [the section called “Siapkan cluster Inferentia”](#).

Topik

- [Jalankan wadah yang dipercepat GPU \(Linux aktif\) EC2](#)
- [Jalankan kontainer yang dipercepat GPU \(Windows pada G-Series EC2 \)](#)
- [Buat grup node terkelola dengan Blok Kapasitas untuk ML](#)
- [Buat node yang dikelola sendiri dengan Blok Kapasitas untuk ML](#)
- [Mencegah Pod agar tidak dijadwalkan pada node tertentu](#)
- [Jalankan pelatihan pembelajaran mesin di Amazon EKS dengan Adaptor Kain Elastis](#)
- [Menggunakan instans AWS Inferentia dengan Amazon EKS untuk Machine Learning](#)

Jalankan wadah yang dipercepat GPU (Linux aktif) EC2

Amazon EKS yang dioptimalkan Amazon Linux AMIs dipercepat dibangun di atas standar Amazon EKS yang dioptimalkan Amazon Linux AMIs. Untuk detail tentang ini AMIs, lihat [the section called “Amazon EKS mengoptimalkan Amazon Linux yang dipercepat AMIs”](#). Teks berikut menjelaskan cara mengaktifkan beban kerja AWS berbasis Neuron.

Untuk mengaktifkan beban kerja berbasis AWS Neuron (akselerator mL)

Untuk detail tentang beban kerja pelatihan dan inferensi menggunakan Neuron di Amazon EKS, lihat referensi berikut:

- [Container - Kubernetes - Memulai](#) Dokumentasi Neuron AWS
- [Pelatihan](#) Sampel AWS Neuron EKS pada GitHub
- [Menerapkan beban kerja inferensi ML dengan di AWSInferentia Amazon EKS](#)

Prosedur berikut menjelaskan cara menjalankan beban kerja pada instance berbasis GPU dengan akselerasi Amazon EKS yang dioptimalkan. AMIs

1. Setelah node GPU Anda bergabung dengan cluster Anda, Anda harus menerapkan [plugin perangkat NVIDIA untuk Kubernetes](#) sebagai a DaemonSet di cluster Anda. Ganti `vX.X.X` dengan s-device-plugin versi [NVIDIA/K8](#) yang Anda inginkan sebelum menjalankan perintah berikut.

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/vX.X.X/
deployments/static/nvidia-device-plugin.yml
```

2. Anda dapat memverifikasi bahwa node Anda telah dialokasikan GPUs dengan perintah berikut.

```
kubectl get nodes "-o=custom-
columns=NAME:.metadata.name,GPU:.status.allocatable.nvidia\.com/gpu"
```

3. Buat file bernama `nvidia-smi.yaml` dengan isi berikut ini. Ganti `tag` dengan tag yang Anda inginkan untuk [nvidia/cuda](#). Manifes ini meluncurkan wadah [NVIDIA CUDA](#) yang berjalan `nvidia-smi` pada node.

```
apiVersion: v1
kind: Pod
metadata:
  name: nvidia-smi
spec:
  restartPolicy: OnFailure
  containers:
  - name: nvidia-smi
    image: nvidia/cuda:tag
    args:
    - "nvidia-smi"
    resources:
      limits:
        nvidia.com/gpu: 1
```

4. Terapkan manifes dengan perintah berikut.

```
kubectl apply -f nvidia-smi.yaml
```

5. Setelah Pod selesai berjalan, lihat lognya dengan perintah berikut.

```
kubectl logs nvidia-smi
```

Contoh output adalah sebagai berikut.

```

Mon Aug 6 20:23:31 20XX
+-----+
| NVIDIA-SMI  XXX.XX                Driver Version:  XXX.XX                |
+-----+-----+-----+-----+-----+-----+
| GPU  Name            Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf    Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
|=====+=====+=====+=====+=====+=====+
|   0   Tesla V100-SXM2...    On   | 00000000:00:1C.0 Off |                    0 |
| N/A   46C    P0     47W / 300W |      0MiB / 16160MiB |           0%      Default |
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
| Processes:                                             GPU Memory |
|  GPU           PID    Type    Process name                     Usage      |
|=====+=====+=====+=====+=====+=====+
| No running processes found                               |
+-----+-----+-----+-----+-----+-----+

```

Jalankan kontainer yang dipercepat GPU (Windows pada G-Series EC2)

Important

[Plugin Perangkat Kubernetes untuk DirectX](#) by TensorWorks adalah alat pihak ketiga yang tidak didukung, didukung, atau dikelola oleh AWS. AWS tidak bertanggung jawab atas keamanan, keandalan, atau kinerja plugin ini.

Pelajari cara menjalankan beban kerja container Windows yang dipercepat GPU di Amazon EKS (Elastic Kubernetes Service) menggunakan NVIDIA GPUs dengan Kubernetes Device Plugin for DirectX by. TensorWorks Untuk informasi selengkapnya, lihat [Kubernetes Device Plugin for DirectX](#).

Ada dua pendekatan utama untuk menyiapkan akselerasi GPU untuk wadah Windows Anda:

- Opsi 1: [Buat AMI EKS Windows Dioptimalkan khusus](#) dengan driver GPU yang diperlukan yang sudah diinstal sebelumnya.
- Gunakan pendekatan ini ketika Anda membutuhkan lingkungan yang konsisten dan telah dikonfigurasi sebelumnya yang siap menjalankan kontainer Windows yang dipercepat GPU, dan Anda dapat menginvestasikan upaya tambahan untuk membangun dan memelihara AMI kustom.

- Opsi 2: Instal driver GPU yang diperlukan pada node pekerja EKS Anda setelah meluncurkan instance Anda.
 - Gunakan pendekatan ini saat Anda menginginkan proses penyiapan yang lebih sederhana dan jangan keberatan menginstal driver GPU di setiap node pekerja baru. Lebih cocok untuk lingkungan pengembangan saat Anda mengevaluasi atau membuat prototipe beban kerja yang dipercepat GPU.

Kedua pendekatan dapat dimanfaatkan menggunakan langkah-langkah yang dirinci dalam panduan ini.

Pertimbangan

Panduan ini menyediakan langkah-langkah untuk menginstal dan mengatur akselerasi GPU untuk container Windows Anda menggunakan NVIDIA GPUs, driver NVIDIA GRID, dan [Plugin Perangkat Kubernetes](#) untuk DirectX by. TensorWorks Langkah-langkah telah diuji dan diverifikasi untuk memberikan akselerasi GPU untuk beban kerja penampung Windows Anda di Amazon EKS. Lihat [the section called “Keterbatasan yang Sudah Diketahui”](#) untuk informasi selengkapnya tentang driver dan plugin perangkat yang kompatibel. Sebelum melanjutkan, perhatikan hal berikut:

- Hanya tipe instans G-family dengan [driver NVIDIA GRID](#) yang telah diuji dan diverifikasi untuk bekerja dengan panduan ini. Sementara jenis instans lain dan kombinasi driver mungkin juga mampu menjalankan kontainer Windows yang dipercepat GPU, mereka mungkin memerlukan langkah-langkah konfigurasi tambahan yang tidak tercakup dalam panduan ini.
- Hanya beban kerja berbasis DirectX yang telah diuji dan diverifikasi untuk bekerja dengan panduan ini. Sementara GPU lain APIs seperti OpenGL, Vulkan, dan OpenCL mungkin berpotensi kompatibel untuk menjalankan wadah Windows yang dipercepat GPU, mereka mungkin memerlukan langkah-langkah konfigurasi tambahan yang tidak tercakup dalam panduan ini.
- Ada beberapa batasan yang diketahui harus diperhatikan sebelum menjalankan wadah Windows yang dipercepat GPU. Silakan lihat [the section called “Keterbatasan yang Sudah Diketahui”](#) bagian untuk informasi lebih lanjut.

Prasyarat

Untuk mengaktifkan akselerasi GPU untuk container Windows Anda di Amazon EKS, Anda harus menyiapkan persyaratan berikut sebelum melanjutkan:

- Luncurkan cluster Amazon EKS dengan Kubernetes v1.27 atau yang lebih baru.

- Menyediakan node Windows dengan Windows Server 2022 atau yang lebih baru.
- [Menyediakan node Windows dalam keluarga G jenis instance, seperti G4 atau G5.](#)
- Menyediakan node Windows dengan runtime container dengan containerd 1.7.x atau 2.x.x (Lihat [the section called “Dapatkan informasi versi”](#) untuk memverifikasi versi containerd di Amazon EKS Optimized AMI Anda.)

Instal driver GPU pada setiap node Windows

Untuk menginstal driver NVIDIA GRID pada node pekerja EKS Anda, ikuti langkah-langkah yang diuraikan dalam [driver NVIDIA untuk EC2 instans Amazon Anda](#). Arahkan ke [opsi Instalasi - Opsi 3: Driver GRID](#) dan ikuti langkah-langkah instalasi.

Instal untuk Windows Server Core

Untuk Windows Server Core, yang tidak memiliki pengalaman desktop, instal driver NVIDIA GRID secara diam-diam dengan menggunakan perintah berikut:

```
$nvidiaInstallerFilePath = nvidia-driver-installer.exe # Replace with path to installer
$installerArguments = "-s -clean -noreboot -noeula"
Start-Process -FilePath $nvidiaInstallerFilePath -ArgumentList $installerArguments -
Wait -NoNewWindow -PassThru
```

Verifikasi instalasi Anda

Jalankan PowerShell perintah berikut untuk menampilkan informasi diagnostik tentang GPUs pada instance:

```
nvidia-smi
```

Perintah ini menampilkan versi driver NVIDIA, serta informasi tentang perangkat keras GPU. Pastikan bahwa output dari perintah ini cocok dengan versi driver NVIDIA GRID yang Anda harapkan akan diinstal.

Menyebarkan plugin perangkat GPU di setiap node

Untuk mengaktifkan penemuan dan eksposur sumber daya GPU ke wadah di node Windows Anda, Anda memerlukan plugin perangkat. Terapkan Plugin [Perangkat DirectX](#) oleh Tensorworks di setiap node pekerja dengan menjalankannya sebagai DaemonSet di cluster EKS Anda. Ikuti panduan

instalasi yang ditentukan dalam [README.md](#), yang akan memerlukan langkah-langkah berikut. Disarankan untuk:

- Menyebarkan plugin perangkat di kube-system namespace.
- Tetapkan batas sumber daya yang sesuai DaemonSet untuk memastikan tidak mengonsumsi sumber daya yang berlebihan pada node Anda.

 Note

Plugin perangkat DaemonSet akan berjalan di setiap node sebagai wadah proses host dengan hak istimewa yang ditinggikan. Disarankan untuk menerapkan kontrol RBAC untuk membatasi akses ke ini DaemonSet sehingga hanya pengguna yang berwenang yang dapat menjalankan perintah istimewa.

Saat menjalankan kontainer yang dipercepat GPU, plugin perangkat mendukung dua mode:

- Mode penyewaan tunggal: Mode ini mendedikasikan semua sumber daya GPU ke satu wadah pada instance. Instal plugin perangkat dengan dukungan penyewaan tunggal menggunakan perintah berikut. Lihat README.md untuk informasi lebih lanjut.

```
kubectl apply -f "https://raw.githubusercontent.com/TensorWorks/directx-device-plugins/main/deployments/default-daemonsets.yml"
```

- Mode multi-tenancy: Mode ini memungkinkan berbagi sumber daya GPU di antara beberapa kontainer pada instance. Instal plugin perangkat dengan dukungan multi-tenancy menggunakan perintah berikut. Lihat README.md untuk informasi lebih lanjut.

```
kubectl apply -f "https://raw.githubusercontent.com/TensorWorks/directx-device-plugins/main/deployments/multitenancy-inline.yml"
```

Atau, gunakan a ConfigMap untuk menentukan multi-tenancy.

```
kubectl apply -f "https://raw.githubusercontent.com/TensorWorks/directx-device-plugins/main/deployments/multitenancy-configmap.yml"
```

Memverifikasi penyebaran plugin perangkat

Setelah Anda menerapkan plugin perangkat, ganti `<namespace>` dan jalankan perintah berikut untuk memverifikasi DirectX Device Plugin berjalan dengan benar di semua node Windows Anda.

```
kubectl get ds device-plugin-wddm -n <namespace>
```

Memverifikasi kontainer siap untuk digunakan

Setelah plugin DaemonSet perangkat berjalan pada node pekerja Windows yang didukung GPU, gunakan perintah berikut untuk memverifikasi bahwa setiap node telah dialokasikan. GPUs Nomor yang sesuai harus sesuai dengan jumlah perangkat DirectX pada setiap node.

```
kubectl get nodes "-o=custom-  
columns=NAME:.metadata.name,DirectX:.status.allocatable.directx\.microsoft\.com/  
display"
```

Menjalankan wadah Windows dengan akselerasi GPU

Sebelum meluncurkan pod Anda, tentukan nama resource `directx.microsoft.com/display` di `di.spec.containers[].resources`. Ini akan menunjukkan bahwa kontainer Anda memerlukan kemampuan berkemampuan GPU, dan `kube-scheduler` akan mencoba menempatkan pod Anda pada node Windows yang telah dikonfigurasi sebelumnya dengan sumber daya GPU yang tersedia.

Sebagai contoh, lihat contoh perintah di bawah ini yang meluncurkan simulasi Job to run Monte Carlo untuk memperkirakan nilai pi. Contoh ini berasal dari [Kubernetes Device Plugins for GitHub DirectX](#) repository, yang [memiliki beberapa](#) contoh untuk dipilih yang dapat Anda jalankan untuk menguji kemampuan GPU node Windows Anda.

```
cat <<EOF | kubectl apply -f -  
apiVersion: batch/v1  
kind: Job  
metadata:  
  name: example-cuda-montecarlo-wddm  
spec:  
  template:  
    spec:  
      containers:  
      - name: example-cuda-montecarlo-wddm  
        image: "index.docker.io/tensorworks/example-cuda-montecarlo:0.0.1"
```

```
resources:
  limits:
    directx.microsoft.com/display: 1
  nodeSelector:
    "kubernetes.io/os": windows
  restartPolicy: Never
  backoffLimit: 0
EOF
```

Keterbatasan yang Sudah Diketahui

Semua GPUs bisa digunakan

Semua GPUs pada instance akan dapat digunakan oleh setiap kontainer yang sedang berjalan di host, bahkan ketika Anda meminta jumlah tertentu GPUs untuk wadah tertentu. Selain itu, perilaku default adalah bahwa semua kontainer yang berjalan di host akan menggunakan GPU dengan indeks 0, bahkan jika ada beberapa yang GPUs tersedia di node. Jadi, agar tugas multi-GPU beroperasi dengan benar, Anda harus secara eksplisit menunjuk perangkat GPU tertentu yang akan digunakan dalam kode aplikasi Anda.

Implementasi yang tepat untuk mengalokasikan perangkat yang akan digunakan untuk aplikasi akan tergantung pada bahasa pemrograman atau kerangka kerja yang Anda gunakan. [Misalnya, jika Anda menggunakan pemrograman CUDA, untuk memilih GPU tertentu, Anda dapat secara eksplisit menentukan perangkat yang akan digunakan dalam kode aplikasi Anda dengan menggunakan fungsi `\(\)`. `cudaSetDevice`](#)

Kebutuhan untuk menentukan perangkat secara eksplisit disebabkan oleh masalah yang diketahui memengaruhi wadah Windows. Anda dapat melacak kemajuan dalam menyelesaikan masalah ini di masalah [Microsoft/Windows-Containers #333](#). Tabel berikut merupakan representasi visual dan contoh praktis dari perilaku alokasi GPU ini.

Pertimbangkan skenario di mana ada satu node Windows tipe EC2 instance `g4dn.12xlarge`, yang dilengkapi dengan empat GPUs. Pertimbangkan skenario di mana tiga pod diluncurkan pada instance ini. Tabel menunjukkan bahwa terlepas dari jumlah yang GPUs diminta oleh setiap kontainer, ketiga pod memiliki akses ke keempatnya GPUs pada instance, dan secara default akan menggunakan GPU dengan indeks perangkat 0.

Pod	Diminta GPUs	Akses GPU Aktual	Penggunaan GPU Default	Indeks GPU yang Tersedia	Jumlah Instance GPUs
Pod 1	1 GPU	Semua 4 GPUs	GPU dengan indeks 0	0, 1, 2, 3	4
Pod 2	2 GPUs	Semua 4 GPUs	GPU dengan indeks 0	0, 1, 2, 3	4
Pod 3	1 GPU	Semua 4 GPUs	GPU dengan indeks 0	0, 1, 2, 3	4

Dukungan plugin perangkat Kubernetes

Implementasi resmi NVIDIA dari [plugin perangkat Kubernetes](#) tidak mendukung Windows. Anda dapat melacak kemajuan dalam menambahkan dukungan Windows resmi di edisi [s-device-plugin NVIDIA/K8](#) #419.

Keterbatasan instans komputasi GPU

Bergantung pada konfigurasi AWS akun, Anda mungkin memiliki batasan layanan pada jumlah dan jenis instans komputasi EC2 GPU Amazon yang dapat Anda luncurkan. Jika Anda membutuhkan kapasitas tambahan, Anda dapat [Meminta kenaikan kuota](#).

Harus membangun AMI yang Dioptimalkan GPU Windows

Tidak ada komponen yang dikelola AKS Windows GPU Optimized AMI atau EC2 Image Builder yang disediakan oleh Amazon EKS. Anda harus mengikuti langkah-langkah dalam panduan ini untuk membuat AMI EKS Windows Optimized kustom dengan driver GPU yang diperlukan yang sudah diinstal sebelumnya, atau menginstal driver GPU yang diperlukan pada node pekerja EKS Anda setelah meluncurkan instance Anda.

Inferensia dan Trainium tidak didukung

AWS Beban kerja berbasis [Inferentia](#) dan AWS [Trainium](#) tidak didukung di Windows.

Buat grup node terkelola dengan Blok Kapasitas untuk ML

Blok Kapasitas untuk pembelajaran mesin (ML) memungkinkan Anda melakukan reservasi instans GPU di masa mendatang untuk mendukung beban kerja ML berdurasi pendek Anda. Untuk informasi selengkapnya, lihat [Blok Kapasitas untuk ML](#) di Panduan EC2 Pengguna Amazon untuk Instans Linux.

Pertimbangan

Important

- Blok Kapasitas hanya tersedia untuk jenis dan AWS Wilayah EC2 instans Amazon tertentu. Untuk informasi kompatibilitas, lihat [Prasyarat Bekerja dengan Blok Kapasitas](#) di Panduan EC2 Pengguna Amazon untuk Instans Linux.
- Untuk informasi selengkapnya, lihat [Menggunakan Blok Kapasitas untuk beban kerja pembelajaran mesin](#) di Panduan Pengguna Amazon EC2 Auto Scaling.
- Grup node terkelola dengan Blok Kapasitas hanya dapat dibuat dengan templat peluncuran khusus.
- Saat memutakhirkan grup node terkelola dengan Blok Kapasitas, pastikan ukuran grup node yang diinginkan diatur ke 0.

Membuat grup node terkelola dengan Amazon EC2 Capacity Blocks

Anda dapat menggunakan Blok Kapasitas dengan grup node terkelola Amazon EKS untuk menyediakan dan menskalakan node pekerja yang dipercepat GPU. Contoh AWS CloudFormation template yang mengikuti tidak mencakup setiap aspek yang diperlukan dalam cluster produksi. Biasanya, Anda juga ingin skrip bootstrap bergabung dengan node ke cluster dan menentukan AMI yang dipercepat Amazon EKS. Untuk informasi selengkapnya, lihat [the section called "Buat"](#).

1. Buat template peluncuran yang sesuai untuk beban kerja Anda dan bekerja dengan grup node terkelola Amazon EKS. Untuk informasi selengkapnya, lihat [the section called "Templat peluncuran"](#).

Selain persyaratan dalam prosedur di atas, pastikan bahwa `LaunchTemplateData` termasuk yang berikut:

- `InstanceMarketOptions` dengan `MarketType` set ke "capacity-block"

- `CapacityReservationSpecification`: `CapacityReservationTarget` dengan `CapacityReservationId` diatur ke Blok Kapasitas (misalnya: `cr-02168da1478b509e0`)
- `InstanceType` disetel ke jenis instance yang mendukung Blok Kapasitas (misalnya: `p5.48xlarge`)

Berikut ini adalah kutipan dari template yang membuat CloudFormation template peluncuran yang menargetkan Blok Kapasitas. Untuk membuat grup node terkelola AMI kustom, Anda juga dapat menambahkan `ImageId` dan `UserData` parameter.

```
NodeLaunchTemplate:
  Type: "AWS::EC2::LaunchTemplate"
  Properties:
    LaunchTemplateData:
      InstanceMarketOptions:
        MarketType: "capacity-block"
      CapacityReservationSpecification:
        CapacityReservationTarget:
          CapacityReservationId: "cr-02168da1478b509e0"
      InstanceType: p5.48xlarge
```

2. Gunakan template peluncuran untuk membuat grup node terkelola.

Berikut ini adalah contoh membuat perintah grup node untuk Blok Kapasitas. Ganti *example-values* dengan yang berlaku untuk cluster Anda.

Saat membuat grup node terkelola Blok Kapasitas, lakukan hal berikut:

- Atur `capacity-type` ke `"CAPACITY_BLOCK"` . Jika jenis kapasitas tidak disetel ke `"CAPACITY_BLOCK"` atau salah satu nilai template peluncuran wajib lainnya di atas hilang, maka permintaan buat akan ditolak.
- Saat menentukan subnets dalam permintaan buat, pastikan untuk hanya menentukan subnet di Availability Zone yang sama dengan reservasi kapasitas.
- Jika Anda menentukan bukan nol `desiredSize` dalam permintaan buat, Amazon EKS akan menghormatinya saat membuat grup Auto Scaling (ASG). Namun, jika permintaan pembuatan dibuat sebelum reservasi kapasitas aktif, ASG tidak akan dapat meluncurkan EC2 instans Amazon hingga menjadi aktif. Akibatnya, aktivitas penskalaan ASG akan mengalami kesalahan peluncuran. Setiap kali reservasi menjadi aktif, maka peluncuran instans akan berhasil dan ASG akan ditingkatkan hingga yang `desiredSize` disebutkan pada waktu pembuatan.

```
aws eks create-nodegroup \
```



```
--cluster-name my-cluster \  
--nodegroup-name my-mng \  
--node-role node-role-arn \  
--region region-code \  
--subnets subnet-id \  
--scaling-config minSize=node-group-min-size,maxSize=node-group-max-  
size,desiredSize=node-group-desired-size \  
--ami-type "AL2023_x86_64_NVIDIA" \  
--capacity-type "CAPACITY_BLOCK" \  
--launch-template id="lt-id",version=1
```

3. Pastikan node bergabung setelah skala naik. Cluster Amazon EKS yang menggunakan grup node dikelola dengan Blok Kapasitas tidak melakukan validasi apa pun bahwa instans yang diluncurkan benar-benar bergabung dan mendaftar dengan cluster.
4. Jika Anda `desiredSize` mengatur 0 pada waktu pembuatan, maka Anda memiliki opsi berbeda untuk meningkatkan grup node saat reservasi kapasitas menjadi aktif:
 - Buat kebijakan penskalaan terjadwal untuk ASG yang selaras dengan waktu mulai reservasi Blok Kapasitas. Untuk informasi selengkapnya, lihat [Penskalaan terjadwal untuk EC2 Auto Scaling Amazon](#) di Panduan Pengguna Amazon Auto EC2 Scaling.
 - Gunakan konsol Amazon EKS atau `eks update-nodegroup-config` untuk memperbarui konfigurasi penskalaan dan mengatur ukuran grup node yang diinginkan.
 - Gunakan Kubernetes Cluster Autoscaler. Untuk informasi selengkapnya, lihat [Cluster Autoscaler](#) di. AWS
5. Grup node sekarang siap untuk beban kerja dan Pod yang akan dijadwalkan.
6. Agar Pod Anda terkuras dengan baik sebelum reservasi berakhir, Amazon EKS menggunakan kebijakan penskalaan terjadwal untuk menurunkan ukuran grup node menjadi 0. Penskalaan terjadwal ini akan ditetapkan dengan nama berjudul Amazon EKS Node Group Capacity Scaledown Before Reservation End. Kami menyarankan untuk tidak mengedit atau menghapus tindakan ini.

Amazon EC2 mulai mematikan instans 30 menit sebelum waktu akhir reservasi. Akibatnya, Amazon EKS akan mengatur penurunan skala terjadwal pada grup node 40 menit sebelum reservasi mereka berakhir untuk mengusir Pod dengan aman dan anggun.

Buat node yang dikelola sendiri dengan Blok Kapasitas untuk ML

Blok Kapasitas untuk pembelajaran mesin (ML) memungkinkan Anda untuk memesan instans GPU di masa mendatang untuk mendukung beban kerja MS berdurasi pendek Anda. Untuk informasi selengkapnya, lihat [Blok Kapasitas untuk ML](#) di Panduan EC2 Pengguna Amazon untuk Instans Linux.

Pertimbangan

Important

- Blok Kapasitas hanya tersedia untuk jenis dan AWS Wilayah EC2 instans Amazon tertentu. Untuk informasi kompatibilitas, lihat [Prasyarat Bekerja dengan Blok Kapasitas di](#) Panduan EC2 Pengguna Amazon untuk Instans Linux.
- Blok Kapasitas saat ini tidak dapat digunakan dengan Karpenter.
- Jika Anda membuat grup node yang dikelola sendiri sebelum reservasi kapasitas menjadi aktif, maka atur kapasitas yang diinginkan. 0
- Untuk memberikan waktu yang cukup untuk menguras node dengan baik, kami sarankan Anda menjadwalkan penskalaan ke skala nol lebih dari 30 menit sebelum waktu akhir reservasi Blok Kapasitas.
- Agar Pod Anda terkuras dengan baik, kami sarankan Anda menyiapkan AWS Node Termination Handler seperti yang dijelaskan dalam contoh langkah.

Gunakan Blok Kapasitas dengan node yang dikelola sendiri

Anda dapat menggunakan Blok Kapasitas dengan Amazon EKS untuk menyediakan dan menskalakan node yang dikelola sendiri. Langkah-langkah berikut memberikan gambaran umum contoh. Contoh AWS CloudFormation template tidak mencakup setiap aspek yang diperlukan dalam beban kerja produksi. Biasanya Anda juga ingin skrip bootstrap bergabung dengan node ke cluster, menentukan AMI yang dipercepat Amazon EKS, dan profil instance yang sesuai untuk bergabung dengan cluster. Untuk informasi selengkapnya, lihat [the section called “Amazon Linux”](#).

1. Buat template peluncuran yang berlaku untuk beban kerja Anda. Untuk informasi selengkapnya, lihat [Menggunakan Blok Kapasitas untuk beban kerja pembelajaran mesin](#) di Panduan Pengguna Amazon EC2 Auto Scaling.

Pastikan yang `LaunchTemplateData` termasuk yang berikut:

- `InstanceMarketOptions` dengan `MarketType` set ke `"capacity-block"`
- `CapacityReservationSpecification`: `CapacityReservationTarget` dengan `CapacityReservationId` diatur ke Blok Kapasitas (misalnya: `cr-02168da1478b509e0`)
- `IamInstanceProfile` dengan `Arn` set ke yang berlaku `iam-instance-profile-arn`
- `ImageId` diatur ke yang berlaku `image-id`
- `InstanceType` disetel ke jenis instance yang mendukung Blok Kapasitas (misalnya: `p5.48xlarge`)
- `SecurityGroupIds` diatur ke yang berlaku IDs (misalnya: `sg-05b1d815d1EXAMPLE`)
- `UserData` disetel ke yang berlaku `user-data` untuk grup node yang dikelola sendiri

Berikut ini adalah kutipan dari template yang membuat CloudFormation template peluncuran yang menargetkan Blok Kapasitas.

```
NodeLaunchTemplate:
  Type: "aws::EC2::LaunchTemplate"
  Properties:
    LaunchTemplateData:
      InstanceMarketOptions:
        MarketType: "capacity-block"
      CapacityReservationSpecification:
        CapacityReservationTarget:
          CapacityReservationId: "cr-02168da1478b509e0"
      IamInstanceProfile:
        Arn: iam-instance-profile-arn
      ImageId: image-id
      InstanceType: p5.48xlarge
      KeyName: key-name
      SecurityGroupIds:
        - sg-05b1d815d1EXAMPLE
      UserData: user-data
```

Anda harus melewati subnet di Availability Zone di mana reservasi dilakukan karena Blok Kapasitas bersifat zonal.

- Gunakan template peluncuran untuk membuat grup node yang dikelola sendiri. Jika Anda melakukan ini sebelum reservasi kapasitas menjadi aktif, maka atur kapasitas yang diinginkan⁰.

Saat membuat grup node, pastikan bahwa Anda hanya menentukan subnet masing-masing untuk Availability Zone di mana kapasitas dicadangkan.

Berikut ini adalah contoh CloudFormation template yang dapat Anda referensi saat membuat satu yang berlaku untuk beban kerja Anda. Contoh ini mendapatkan LaunchTemplateId dan Version dari `AWS::Amazon::EC2::LaunchTemplate` sumber daya yang ditunjukkan pada langkah sebelumnya. Itu juga mendapatkan nilai untuk `DesiredCapacity`, `MaxSize`, `MinSize`, dan `VPCZoneIdentifier` yang dideklarasikan di tempat lain dalam template yang sama.

```
NodeGroup:
  Type: "AWS::AutoScaling::AutoScalingGroup"
  Properties:
    DesiredCapacity: !Ref NodeAutoScalingGroupDesiredCapacity
    LaunchTemplate:
      LaunchTemplateId: !Ref NodeLaunchTemplate
      Version: !GetAtt NodeLaunchTemplate.LatestVersionNumber
    MaxSize: !Ref NodeAutoScalingGroupMaxSize
    MinSize: !Ref NodeAutoScalingGroupMinSize
    VPCZoneIdentifier: !Ref Subnets
  Tags:
    - Key: Name
      PropagateAtLaunch: true
      Value: !Sub ${ClusterName}-${NodeGroupName}-Node
    - Key: !Sub kubernetes.io/cluster/${ClusterName}
      PropagateAtLaunch: true
      Value: owned
```

3. Setelah grup node berhasil dibuat, pastikan untuk merekam `NodeInstanceRole` untuk grup node yang dibuat. Anda memerlukan ini untuk memastikan bahwa ketika grup node diskalakan, node baru bergabung dengan cluster dan Kubernetes dapat mengenali node. Untuk informasi selengkapnya, lihat AWS Management Console petunjuk di [Membuat node Amazon Linux yang dikelola sendiri](#).
4. Kami menyarankan Anda membuat kebijakan penskalaan terjadwal untuk grup Auto Scaling yang sejajar dengan waktu reservasi Blok Kapasitas. Untuk informasi selengkapnya, lihat [Penskalaan terjadwal untuk EC2 Auto Scaling Amazon](#) di Panduan Pengguna Amazon Auto EC2 Scaling.

Anda dapat menggunakan semua instans yang Anda pesan hingga 30 menit sebelum waktu akhir Blok Kapasitas. Contoh yang masih berjalan pada saat itu akan mulai berakhir. Untuk memberikan waktu yang cukup untuk menguras node dengan baik, kami sarankan Anda menjadwalkan penskalaan ke skala nol lebih dari 30 menit sebelum waktu akhir reservasi Blok Kapasitas.

Jika Anda ingin meningkatkan secara manual setiap kali reservasi kapasitas terjadiActive, maka Anda perlu memperbarui kapasitas yang diinginkan grup Auto Scaling pada waktu mulai reservasi Blok Kapasitas. Kemudian Anda juga perlu menurunkan skala secara manual lebih dari 30 menit sebelum waktu akhir reservasi Blok Kapasitas.

5. Grup node sekarang siap untuk beban kerja dan Pod yang akan dijadwalkan.
6. Agar Pod Anda terkuras dengan baik, kami sarankan Anda menyiapkan AWS Node Termination Handler. Handler ini akan dapat mengawasi peristiwa siklus hidup “ASG Scale-in” dari Amazon Auto EC2 Scaling EventBridge menggunakan dan memungkinkan bidang kontrol Kubernetes untuk mengambil tindakan yang diperlukan sebelum instance menjadi tidak tersedia. Jika tidak, objek Pod dan Kubernetes Anda akan macet dalam status tertunda. Untuk informasi selengkapnya, lihat [AWS Node Termination Handler](#) di GitHub.

Jika Anda tidak menyiapkan Node Termination Handler, sebaiknya Anda mulai menguras Pod secara manual sebelum menekan jendela 30 menit sehingga mereka memiliki cukup waktu untuk terkuras dengan baik.

Mencegah Pod agar tidak dijadwalkan pada node tertentu

Node dengan prosesor khusus, seperti GPUs, bisa lebih mahal untuk dijalankan daripada node yang berjalan pada mesin yang lebih standar. Untuk alasan itu, Anda mungkin ingin melindungi node tersebut dari beban kerja yang tidak memerlukan perangkat keras khusus agar tidak digunakan ke node tersebut. Salah satu cara untuk melakukannya adalah dengan noda.

Amazon EKS mendukung konfigurasi noda Kubernetes melalui grup simpul terkelola. Taint dan toleransi bekerja sama untuk memastikan bahwa Pod tidak dijadwalkan ke node yang tidak sesuai. Satu atau lebih cacat dapat diterapkan pada sebuah simpul. Ini menandai bahwa node tidak boleh menerima Pod apa pun yang tidak mentolerir taints. Toleransi diterapkan pada Pod dan memungkinkan, tetapi tidak mengharuskan, Pod untuk menjadwalkan ke node dengan taints yang cocok. Untuk informasi selengkapnya, lihat [Taints and Tolerations dalam dokumentasi](#) Kubernetes.

Node taints Kubernetes dapat diterapkan ke grup node terkelola baru dan yang sudah ada menggunakan atau AWS Management Console melalui Amazon EKS API.

- Untuk informasi tentang membuat grup node dengan taint menggunakan AWS Management Console, lihat [the section called “Buat”](#).
- Berikut ini adalah contoh membuat grup node dengan taint menggunakan AWS CLI:

```
aws eks create-nodegroup \
--cli-input-json '
{
  "clusterName": "my-cluster",
  "nodegroupName": "node-taints-example",
  "subnets": [
    "subnet-1234567890abcdef0",
    "subnet-abcdef01234567890",
    "subnet-021345abcdef67890"
  ],
  "nodeRole": "arn:aws:iam::111122223333:role/AmazonEKSNodeRole",
  "taints": [
    {
      "key": "dedicated",
      "value": "gpuGroup",
      "effect": "NO_SCHEDULE"
    }
  ]
}'
```

Untuk informasi selengkapnya dan contoh penggunaan, lihat [taint di dokumentasi](#) referensi Kubernetes.

Note

- Taints dapat diperbarui setelah Anda membuat grup node menggunakan `UpdateNodegroupConfig` API.
- Kunci cacat harus diawali dengan huruf atau angka. Ini dapat berisi huruf, angka, tanda hubung (-), periode (.), dan garis bawah (_). Panjangnya bisa mencapai 63 karakter.
- Secara opsional, kunci cacat dapat dimulai dengan prefiks subdomain DNS dan satu /. Jika dimulai dengan prefiks subdomain DNS, panjangnya bisa mencapai 253 karakter.
- Nilai adalah opsional dan harus diawali dengan huruf atau angka. Ini dapat berisi huruf, angka, tanda hubung (-), periode (.), dan garis bawah (_). Panjangnya bisa mencapai 63 karakter.
- Saat menggunakan Kubernetes secara langsung atau AWS Management Console, efek taint harus `NoSchedule`, atau `PreferNoSchedule` `NoExecute` Namun, saat

menggunakan AWS CLI atau API, efek taint harus `NO_SCHEDULE`, `PREFER_NO_SCHEDULE` atau `NO_EXECUTE`

- Maksimal 50 taint diperbolehkan per grup node.
- Jika taint yang dibuat menggunakan grup node terkelola dihapus secara manual dari node, maka Amazon EKS tidak menambahkan taints kembali ke node. Ini benar bahkan jika taints ditentukan dalam konfigurasi grup node terkelola.

Anda dapat menggunakan perintah [aws eks update-nodegroup-config](#) AWS CLI untuk menambahkan, menghapus, atau mengganti taints untuk grup node terkelola.

Jalankan pelatihan pembelajaran mesin di Amazon EKS dengan Adaptor Kain Elastis

Topik ini menjelaskan cara mengintegrasikan Elastic Fabric Adapter (EFA) dengan Pod yang di-deploy di cluster Amazon EKS Anda. Elastic Fabric Adapter (EFA) adalah antarmuka jaringan untuk EC2 instans Amazon yang memungkinkan Anda menjalankan aplikasi yang membutuhkan komunikasi antar-simpul tingkat tinggi dalam skala besar. AWS ini merupakan sistem operasi antarmuka bypass hardware yang meningkatkan performa komunikasi antar-instans, yang sangat penting untuk skala aplikasi ini. Dengan EFA, aplikasi High Performance Computing (HPC) menggunakan aplikasi Message Passing Interface (MPI) dan Machine Learning (ML) menggunakan NVIDIA Collective Communications Library (NCCL) dapat menskalakan hingga ribuan atau. CPUs GPUs Hasilnya, Anda mendapatkan kinerja aplikasi klaster HPC lokal dengan elastisitas dan fleksibilitas cloud sesuai permintaan. AWS Mengintegrasikan EFA dengan aplikasi yang berjalan di klaster Amazon EKS dapat mengurangi waktu untuk menyelesaikan beban kerja pelatihan yang terdistribusi pada skala besar tanpa harus menambahkan instans tambahan terhadap klaster Anda. Untuk informasi selengkapnya, lihat [Elastic Fabric Adaptor](#).

Jenis instans dengan EFA

Plugin Perangkat AWS Kubernetes EFA mendukung semua jenis EC2 instans Amazon yang memiliki EFA. Untuk melihat daftar semua jenis instans yang memiliki EFA, lihat [Jenis instans yang didukung](#) di Panduan EC2 Pengguna Amazon. Namun, untuk menjalankan aplikasi ML dengan cepat, kami merekomendasikan bahwa sebuah instance memiliki chip akselerasi perangkat keras seperti NVidia GPUs, chip [AWS Inferentia](#), atau chip [AWS Trainium](#), selain EFA. Untuk melihat daftar jenis instans yang memiliki chip akselerasi perangkat keras dan EFA, lihat [Komputasi yang dipercepat](#) di Panduan EC2 Pengguna Amazon.

Saat Anda membandingkan jenis instance untuk dipilih di antara mereka, pertimbangkan jumlah kartu jaringan EFA yang tersedia untuk jenis instance tersebut serta jumlah kartu akselerator, jumlah CPU, dan jumlah memori. Anda dapat menetapkan hingga satu EFA per kartu jaringan. EFA dihitung sebagai antarmuka jaringan.. Untuk melihat berapa banyak EFA yang tersedia untuk setiap jenis instans yang memiliki EFA, lihat daftar [Kartu jaringan](#) di EC2 Panduan Pengguna Amazon.

Antarmuka khusus EFA dan EFA

Elastic Fabric Adapter (EFA) adalah antarmuka jaringan yang menggabungkan kemampuan Elastic Network Adapter (ENA) dan antarmuka OS-bypass, didukung oleh protokol AWS Scalable Reliable Datagram (SRD). Fungsionalitas EFA memungkinkan aplikasi untuk berkomunikasi langsung dengan perangkat keras untuk transportasi latensi rendah. Anda dapat memilih untuk mengakses hanya kemampuan EFA menggunakan antarmuka khusus EFA, membatasi komunikasi ke antarmuka dalam Availability Zone yang sama.

Untuk membuat node yang dapat memiliki antarmuka khusus EFA, Anda harus menggunakan Template EC2 Peluncuran kustom dan mengatur ke `InterfaceType efa-only` Di Template Peluncuran kustom Anda, Anda tidak dapat mengatur kartu jaringan `0` ke antarmuka khusus EFA, karena itu adalah kartu jaringan utama dan antarmuka jaringan instance. EC2 Anda harus memiliki versi VPC CNI `1.18.5` atau yang lebih baru untuk antarmuka khusus EFA. Jika Anda menggunakan Amazon Linux 2, versi ami harus `v20240928` atau lebih baru untuk antarmuka khusus EFA.

Prosedur berikut memandu Anda untuk membuat cluster EKS `eksctl` dengan node yang memiliki antarmuka NVidia GPUs dan EFA. Anda tidak dapat menggunakan `eksctl` untuk membuat node dan grup node yang menggunakan antarmuka khusus EFA.

Prasyarat

- Sebuah klaster Amazon EKS yang sudah ada. Jika Anda tidak memiliki cluster yang ada, buat satu menggunakan [Memulai](#) .. Klaster Anda harus men-deploy pada VPC yang memiliki setidaknya satu subnet privat dengan memiliki cukup alamat IP yang tersedia untuk men-deploy pada simpul. Subnet privat harus memiliki akses internet luar yang disediakan oleh perangkat eksternal, seperti gateway NAT.

Jika Anda berencana untuk menggunakan `eksctl` untuk membuat grup node Anda, juga `eksctl` dapat membuat cluster untuk Anda.

- Versi `2.12.3` atau yang lebih baru atau versi `1.27.160` atau yang lebih baru dari AWS Command Line Interface (AWS CLI) diinstal dan dikonfigurasi pada perangkat Anda atau. AWS CloudShell

Untuk memeriksa versi Anda saat ini, gunakan `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Package manager seperti `yum` atau `apt-get`, atau Homebrew untuk macOS seringkali merupakan beberapa versi di belakang versi terbaru CLI. AWS Untuk menginstal versi terbaru, lihat [Menginstal](#) dan [Konfigurasi cepat dengan aws configure](#) di Panduan Pengguna Antarmuka Baris AWS Perintah. Versi AWS CLI yang diinstal AWS CloudShell mungkin juga beberapa versi di belakang versi terbaru. Untuk memperbaruinya, lihat [Menginstal AWS CLI ke direktori home Anda](#) di AWS CloudShell Panduan Pengguna.

- Alat baris perintah `kubectl` diinstal pada perangkat Anda atau AWS CloudShell. Versinya bisa sama dengan atau hingga satu versi minor lebih awal atau lebih lambat dari versi Kubernetes dari kluster Anda. Misalnya, jika versi cluster Anda `1.29`, Anda dapat menggunakan `kubectl` versi `1.28`, `1.29`, atau `1.30` dengan itu. Untuk menginstal atau memutakhirkan `kubectl`, lihat [the section called “Siapkan kubectl dan eksctl”](#).
- Anda harus memiliki plugin Amazon VPC CNI untuk versi Kubernetes `1.7.10` atau yang lebih baru diinstal sebelum meluncurkan node pekerja yang mendukung beberapa Adaptor Kain Elastis, seperti `eni`. Untuk informasi selengkapnya tentang memperbarui plugin Amazon VPC CNI untuk versi Kubernetes, lihat [the section called “Amazon VPC CNI”](#).

Important

Sebuah pertimbangan penting yang diperlukan untuk mengadopsi EFA dengan Kubernetes untuk mengkonfigurasi dan mengelola Huge Pages sebagai sumber daya dalam kluster. Untuk informasi selengkapnya, lihat [Mengelola Halaman Besar](#) dalam dokumentasi Kubernetes. EC2 Instans Amazon dengan driver EFA yang diinstal sebelumnya mengalokasikan 5128 2MiB Halaman Besar, yang dapat Anda minta sebagai sumber daya untuk digunakan dalam spesifikasi pekerjaan Anda.

Buat grup simpul

Prosedur berikut membantu Anda membuat grup node dengan grup node yang `p4d.24xlarge` didukung dengan antarmuka EFA dan GPUDirect RDMA, dan menjalankan contoh pengujian NVIDIA Collective Communications Library (NCCL) untuk menggunakan Kinerja NCCL multi-node. EFAs Contoh ini dapat digunakan template untuk pelatihan pembelajaran mendalam terdistribusi di Amazon EKS menggunakan EFAs.

1. Tentukan jenis EC2 instans Amazon yang mendukung EFA yang tersedia di AWS Wilayah tempat Anda ingin menerapkan node. Ganti *region-code* dengan AWS Region tempat Anda ingin menyebarkan grup node Anda.

```
aws ec2 describe-instance-types --region region-code \  
  --filters Name=network-info.efa-supported,Values=true \  
  --query "InstanceTypes[*].[InstanceType]" --output text
```

Saat Anda menerapkan node, jenis instance yang ingin Anda terapkan harus tersedia di AWS Wilayah tempat kluster Anda berada.

2. Tentukan Availability Zone tempat tipe instans yang ingin Anda gunakan tersedia. Dalam tutorial ini, jenis *p5.48xlarge* instance digunakan dan harus dikembalikan dalam output untuk AWS Wilayah yang Anda tentukan pada langkah sebelumnya. Saat Anda menerapkan node di cluster produksi, ganti *p5.48xlarge* dengan jenis instance apa pun yang dikembalikan pada langkah sebelumnya.

```
aws ec2 describe-instance-type-offerings --region region-code \  
  --location-type availability-zone --filters Name=instance-  
type,Values=p4d.24xlarge,p5.48xlarge \  
  --query 'InstanceTypeOfferings[*].Location' --output text
```

Contoh output adalah sebagai berikut.

```
us-west-2a    us-west-2c    us-west-2b
```

Perhatikan Availability Zones yang dikembalikan untuk digunakan di langkah selanjutnya. Saat Anda menyebarkan node ke cluster, VPC Anda harus memiliki subnet dengan alamat IP yang tersedia di salah satu Availability Zone yang dikembalikan dalam output.

3. Buat grup simpul menggunakan `eksctl`. Anda memerlukan versi `0.210.0` atau yang lebih baru dari alat baris perintah yang diinstal pada perangkat Anda atau AWS CloudShell. Untuk menginstal atau memperbarui `eksctl`, lihat [Instalasi](#) dalam `eksctl` dokumentasi.
 - a. Salin isi berikut ke file bernama *efa-cluster.yaml*. Ganti *example values* dengan milik Anda sendiri. Anda dapat mengganti *p5.48xlarge* dengan instance yang berbeda, tetapi jika Anda melakukannya, pastikan bahwa nilai untuk `availabilityZones` adalah Availability Zones yang dikembalikan untuk jenis instance di langkah 1.

```
apiVersion: eksctl.io/v1alpha5
```

```
kind: ClusterConfig

metadata:
  name: my-efa-cluster
  region: region-code
  version: "1.XX"

iam:
  withOIDC: true

availabilityZones: ["us-west-2a", "us-west-2c"]

managedNodeGroups:
  - name: my-efa-ng
    instanceType: p5.48xlarge
    minSize: 1
    desiredCapacity: 2
    maxSize: 3
    availabilityZones: ["us-west-2a"]
    volumeSize: 300
    privateNetworking: true
    efaEnabled: true
```

b. Membuat grup simpul yang dikelola dalam kluster yang sudah ada.

```
eksctl create nodegroup -f efa-cluster.yaml
```

Jika Anda tidak memiliki cluster yang ada, Anda dapat menjalankan perintah berikut untuk membuat cluster dan grup node.

```
eksctl create cluster -f efa-cluster.yaml
```

Note

Karena tipe instance yang digunakan dalam contoh ini memiliki GPUs, eksctl secara otomatis menginstal plugin perangkat NVIDIA Kubernetes pada setiap instance untuk Anda saat menggunakan Amazon Linux 2. Ini tidak diperlukan untuk Bottlerocket, karena plugin perangkat NVIDIA dibangun ke dalam varian EKS NVIDIA Bottlerocket. Ketika efaEnabled diatur ke true dalam konfigurasi nodegroup, juga eksctl akan secara otomatis menyebarkan plugin perangkat EFA pada node.

Menggunakan Bottlerocket dengan EFA

Bottlerocket AMI versi 1.28.0 dan yang lebih baru termasuk dukungan resmi untuk EFA. Untuk menggunakan Bottlerocket untuk node yang mendukung EFA, tentukan dalam konfigurasi Anda. `amiFamily`: Bottlerocket Jika Anda perlu menggunakan ID AMI kustom, Anda harus menggunakan standar `nodeGroups` alih-alih `managedNodeGroups`.

Berikut adalah contoh konfigurasi:

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-efa-bottlerocket-cluster
  region: region-code
  version: "1.XX"

iam:
  withOIDC: true

availabilityZones: ["us-west-2a", "us-west-2c"]

managedNodeGroups:
- name: my-efa-bottlerocket-ng
  instanceType: p5.48xlarge
  minSize: 1
  desiredCapacity: 2
  maxSize: 3
  availabilityZones: ["us-west-2a"]
  volumeSize: 300
  privateNetworking: true
  efaEnabled: true
  amiFamily: Bottlerocket
  bottlerocket:
    enableAdminContainer: true
    settings:
      kernel:
        sysctl:
          "vm.nr_hugepages": "3000" # Configures 3000 * 2Mi = 6000Mi hugepages
```

Pengaturan `vm.nr_hugepages` `sysctl` di atas mengonfigurasi jumlah halaman besar 2Mi. Dalam contoh ini, 3000 berarti $3000 * 2\text{Mi} = 6000\text{Mi}$ halaman besar.

Verifikasi instalasi plugin perangkat EFA

Ketika Anda membuat grup node dengan `efaEnabled: true`, `eksctl` secara otomatis menyebarkan plugin perangkat EFA Kubernetes untuk Anda. Anda dapat memverifikasi bahwa plugin perangkat diinstal dan berfungsi dengan benar:

1. Periksa DaemonSet statusnya:

```
kubectl get daemonsets -n kube-system
```

Contoh output:

NAME	DESIRED	CURRENT	READY	UP-TO-DATE
aws-efa-k8s-device-plugin-daemonset	2	2	2	2
AVAILABLE NODE SELECTOR AGE				
<none> 6m16s				
...				

Di sini, plugin DaemonSet perangkat EFA berjalan pada dua node. Keduanya sudah siap dan tersedia.

2. Selanjutnya, verifikasi pod yang dibuat oleh DaemonSet:

```
kubectl get pods -n kube-system -l name=aws-efa-k8s-device-plugin
```

Contoh output:

NAME	READY	STATUS	RESTARTS	AGE
aws-efa-k8s-device-plugin-daemonset-d68bs	1/1	Running	0	6m16s
aws-efa-k8s-device-plugin-daemonset-w4l8t	1/1	Running	0	6m16s

Pod plugin perangkat EFA berada dalam status Running, mengonfirmasi bahwa plugin berhasil digunakan dan dioperasikan.

3. Verifikasi pendaftaran sumber daya:

Anda dapat mengonfirmasi bahwa `vpc.amazonaws.com/efa` sumber daya terdaftar dengan `kubelet` dengan menjelaskan node:

```
kubectl describe nodes
```

Jika sumber daya EFA terdaftar dengan benar, Anda akan melihatnya terdaftar di bawah Capacity and Allocatable resources node. Misalnya:

```
Capacity:
...
vpc.amazonaws.com/efa: 4
Allocatable:
...
vpc.amazonaws.com/efa: 4
```

Output ini mengonfirmasi bahwa node mengenali sumber daya EFA, membuatnya tersedia untuk pod yang memintanya.

(Opsional) Uji kinerja EFA

Kami menyarankan Anda menguji pengaturan EFA. Anda dapat menggunakan [Tes NCCL di aws-samples/awesome-distributed-training repository](#) pada GitHub [Tes NCCL](#) mengevaluasi kinerja jaringan menggunakan Perpustakaan Komunikasi Kolektif Nvidia. Langkah-langkah berikut mengirimkan tes NCCL di Amazon EKS.

1. Menerapkan Operator MPI Kubeflow:

Untuk tes NCCL Anda dapat menerapkan Operasi Kubeflow MPI. Operasi MPI mempermudah untuk menjalankan pelatihan terdistribusi tipe-Allreduce di Kubernetes. Untuk informasi lebih lanjut, lihat [Operator MPI](#) di GitHub.

2. Jalankan Uji Kinerja NCCL multi-node untuk memverifikasi RDMA/EFA: GPUDirect

Untuk memverifikasi kinerja NCCL dengan GPUDirect RDMA melalui EFA, jalankan uji Kinerja NCCL standar. Untuk informasi lebih lanjut, lihat repo [NCCL-Tests](#) resmi di GitHub

Selesaikan langkah-langkah berikut ini untuk menjalankan dua simpul uji kinerja NCCL.

Dalam contoh pekerjaan pengujian NCCL, setiap pekerja meminta delapan, 5210Mi GPUs, empagepages-2Mi, dan 8000Mi memori EFAs, yang secara efektif berarti setiap pekerja mengkonsumsi semua sumber daya dari sebuah instance. p5.48xlarge

a. Buat MPIJob manifes:

Salin berikut ini ke file bernama `nccl-tests.yaml`:

```

apiVersion: kubeflow.org/v2beta1
kind: MPIJob
metadata:
  name: nccl-tests
spec:
  runPolicy:
    cleanPodPolicy: Running
    backoffLimit: 20
  slotsPerWorker: 8
  mpiReplicaSpecs:
    Launcher:
      replicas: 1
      template:
        spec:
          restartPolicy: OnFailure
          containers:
            - image: public.ecr.aws/hpc-cloud/nccl-tests:latest
              imagePullPolicy: IfNotPresent
              name: test-nccl-launcher
              env:
                - name: PATH
                  value: $PATH:/opt/amazon/efa/bin:/usr/bin
                - name: LD_LIBRARY_PATH
                  value: /opt/amazon/openmpi/lib:/opt/nccl/build/lib:/opt/amazon/efa/
lib:/opt/aws-ofi-nccl/install/lib:/usr/local/nvidia/lib:$LD_LIBRARY_PATH
                - name: NCCL_DEBUG
                  value: INFO
                - name: NCCL_BUFFSIZE
                  value: '8388608'
                - name: NCCL_P2P_NET_CHUNKSIZE
                  value: '524288'
                - name: NCCL_TUNER_PLUGIN
                  value: /opt/aws-ofi-nccl/install/lib/libnccl-ofi-tuner.so
          command:
            - /opt/amazon/openmpi/bin/mpirun
            - --allow-run-as-root
            - --tag-output
            - -np
            - "16"
            - -N
            - "8"
            - --bind-to
            - none

```

```

- -x
- PATH
- -x
- LD_LIBRARY_PATH
- -x
- NCCL_DEBUG=INFO
- -x
- NCCL_BUFFSIZE
- -x
- NCCL_P2P_NET_CHUNKSIZE
- -x
- NCCL_TUNER_PLUGIN
- --mca
- pml
- ^cm,ucx
- --mca
- btl
- tcp,self
- --mca
- btl_tcp_if_exclude
- lo,docker0,veth_def_agent
- /opt/nccl-tests/build/all_reduce_perf
- -b
- "8"
- -e
- "16G"
- -f
- "2"
- -g
- "1"
- -c
- "1"
- -n
- "100"

```

Worker:

replicas: 2

template:

spec:

nodeSelector:

node.kubernetes.io/instance-type: "p5.48xlarge"

containers:

```

- image: public.ecr.aws/hpc-cloud/nccl-tests:latest
  imagePullPolicy: IfNotPresent
  name: nccl-tests-worker

```



```

volumeMounts:
- name: shm
  mountPath: /dev/shm
resources:
  limits:
    nvidia.com/gpu: 8
    hugepages-2Mi: 5120Mi
    vpc.amazonaws.com/efa: 32
    memory: 32000Mi
  requests:
    nvidia.com/gpu: 8
    hugepages-2Mi: 5120Mi
    vpc.amazonaws.com/efa: 32
    memory: 32000Mi
volumes:
- name: shm
  hostPath:
    path: /dev/shm

```

b. Terapkan MPIJob NCCL-Tests:

Kirim MPIJob dengan menerapkan manifes. Ini akan membuat dua EC2 contoh p5.48xlarge Amazon.

```
kubectl apply -f nccl-tests.yaml
```

Contoh output adalah sebagai berikut.

```
mpijob.kubeflow.org/nccl-tests created
```

c. Verifikasi bahwa pekerjaan memulai pod:

Lihat Pod yang sedang berjalan.

```
kubectl get pods
```

Contoh output adalah sebagai berikut.

NAME	READY	STATUS	RESTARTS	AGE
nccl-tests-launcher-nbql9	0/1	Init:0/1	0	2m49s
nccl-tests-worker-0	1/1	Running	0	2m49s

nccl-tests-worker-1	1/1	Running	0	2m49s

Operator MPI membuat Pod peluncur dan 2 Pod pekerja (satu di setiap node).

d. Verifikasi bahwa pekerjaan berjalan dengan sukses dengan log:

Lihat log untuk `nccl-tests-launcher` Pod. Ganti `nbq19` dengan nilai dari output Anda.

```
kubectl logs -f nccl-tests-launcher-nbq19
```

Jika tes berhasil diselesaikan, Anda dapat menyebarkan aplikasi Anda yang menggunakan Perpustakaan Komunikasi Kolektif Nvidia.

Menggunakan instans AWS Inferentia dengan Amazon EKS untuk Machine Learning

Topik ini menjelaskan cara membuat klaster Amazon EKS dengan node yang menjalankan instans [Amazon EC2 Inf1](#) dan (opsional) menerapkan aplikasi sampel. Instans Amazon EC2 Inf1 didukung oleh chip [AWS Inferentia](#), yang dibuat khusus AWS untuk memberikan kinerja tinggi dan inferensi biaya terendah di cloud. Model pembelajaran mesin digunakan ke wadah menggunakan [AWS Neuron](#), kit pengembangan perangkat lunak khusus (SDK) yang terdiri dari kompiler, runtime, dan alat profil yang mengoptimalkan kinerja inferensi pembelajaran mesin chip Inferentia. AWS Neuron mendukung kerangka kerja pembelajaran mesin populer seperti TensorFlow, PyTorch, dan MXNet.

Note

Perangkat neuron logis IDs harus bersebelahan. Jika sebuah Pod yang meminta beberapa perangkat Neuron dijadwalkan pada tipe `inf1.6xlarge` atau `inf1.24xlarge` instance (yang memiliki lebih dari satu perangkat Neuron), Pod tersebut akan gagal untuk memulai jika penjadwal Kubernetes memilih perangkat yang tidak bersebelahan. IDs Untuk informasi selengkapnya, lihat [Logika perangkat IDs harus bersebelahan](#). GitHub

Prasyarat

- Miliki `eksctl` yang terinstal pada komputer Anda. Jika Anda belum menginstalnya, lihat [Instalasi](#) di `eksctl` dokumentasi.

- Miliki `kubectl` yang ter-install pada komputer Anda. Untuk informasi selengkapnya, lihat [the section called “Siapkan kubectl dan eksctl”](#).
- (Opsional) Miliki `python3` yang ter-install pada komputer Anda. Jika Anda belum menginstalnya, lihat [unduh Python](#) untuk petunjuk penginstalan.

Membuat klaster

1. Buat cluster dengan node EC2 instance `Inf1` Amazon. Anda dapat mengganti *`inf1.2xlarge`* dengan [jenis instans Inf1](#) apa pun. `eksctl`Utilitas mendeteksi bahwa Anda meluncurkan grup node dengan tipe `Inf1` instans dan akan memulai node Anda menggunakan salah satu Amazon EKS yang dioptimalkan Amazon Linux AMIs yang dipercepat.

Note

Anda tidak dapat menggunakan [peran IAM untuk akun layanan](#) dengan TensorFlow Serving.

```
eksctl create cluster \  
  --name inferentia \  
  --region region-code \  
  --nodegroup-name ng-inf1 \  
  --node-type inf1.2xlarge \  
  --nodes 2 \  
  --nodes-min 1 \  
  --nodes-max 4 \  
  --ssh-access \  
  --ssh-public-key your-key \  
  --with-oidc
```

Note

Perhatikan nilai baris output berikut. Ini digunakan dalam langkah selanjutnya (opsional).

```
[9] adding identity "arn:aws:iam::111122223333:role/eksctl-inferentia-nodegroup-ng-  
in-NodeInstanceRole-FI7HIYS3BS09" to auth ConfigMap
```

Saat meluncurkan grup node dengan Inf1 instance, `eksctl` secara otomatis menginstal plugin perangkat AWS Neuron Kubernetes. Plugin ini mengiklankan perangkat Neuron sebagai sumber daya sistem ke penjadwal Kubernetes, yang dapat diminta oleh kontainer. Sebagai tambahan untuk kebijakan IAM simpul Amazon EKS default, Amazon S3 hanya membaca kebijakan akses yang ditambahkan sehingga aplikasi sampel, dibahas dalam langkah berikutnya, dapat memuat model terlatih dari Amazon S3.

2. Pastikan semua Pod sudah dimulai dengan benar.

```
kubectl get pods -n kube-system
```

Output yang disingkat:

NAME	READY	STATUS	RESTARTS	AGE
[...]				
neuron-device-plugin-daemonset-6djhp	1/1	Running	0	5m
neuron-device-plugin-daemonset-hwjsj	1/1	Running	0	5m

(Opsional) Menyebarkan gambar aplikasi TensorFlow Penyajian

Sebuah model terlatih harus dikompilasikan ke target Inferentia sebelum dapat di-deploy pada instans Inferentia. Untuk melanjutkan, Anda memerlukan TensorFlow model yang [dioptimalkan Neuron](#) yang disimpan di Amazon S3. Jika Anda belum memilikinya SavedModel, ikuti tutorial untuk [membuat model ResNet 50 yang kompatibel dengan Neuron](#) dan unggah hasilnya SavedModel ke S3. ResNet-50 adalah model pembelajaran mesin populer yang digunakan untuk tugas pengenalan gambar. Untuk informasi lebih lanjut tentang menyusun model Neuron, lihat [Chip AWS Inferentia Dengan DLAMI di Panduan Pengembang Pembelajaran](#) Mendalam. AWS AMIs

Manifes penerapan sampel mengelola wadah penyajian inferensi bawaan yang TensorFlow disediakan oleh AWS Deep Learning Containers. Di dalam wadah adalah AWS Neuron Runtime dan aplikasi TensorFlow Serving. Daftar lengkap Deep Learning Containers pra-bangun yang dioptimalkan untuk Neuron dipertahankan di GitHub bawah [Gambar yang Tersedia](#). Saat start-up, DLC akan mengambil model Anda dari Amazon S3, meluncurkan Neuron TensorFlow Serving dengan model yang disimpan, dan menunggu permintaan prediksi.

Jumlah perangkat Neuron yang dialokasikan untuk aplikasi penyajian Anda agar dapat disesuaikan dengan mengubah sumber daya `aws.amazon.com/neuron` dalam deployment yaml. Harap dicatat

bahwa komunikasi antara TensorFlow Serving dan runtime Neuron terjadi melalui GRPC, yang membutuhkan penerusan IPC_LOCK kemampuan ke wadah.

1. Tambahkan kebijakan AmazonS3ReadOnlyAccess IAM ke peran instance node yang dibuat pada langkah 1 dari [Buat klaster](#). Hal ini diperlukan agar aplikasi sampel dapat memuat model terlatih dari Amazon S3.

```
aws iam attach-role-policy \  
  --policy-arn arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess \  
  --role-name eksctl-inferentia-nodegroup-ng-in-NodeInstanceRole-FI7HIYS3BS09
```

2. Buat file bernama `rn50_deployment.yaml` dengan isi berikut ini. Memperbarui kode wilayah dan jalur model agar sesuai dengan pengaturan yang Anda inginkan. Nama model adalah untuk tujuan identifikasi ketika klien membuat permintaan ke TensorFlow server. Contoh ini menggunakan nama model untuk mencocokkan contoh skrip klien ResNet 50 yang akan digunakan pada langkah selanjutnya untuk mengirim permintaan prediksi.

```
aws ecr list-images --repository-name neuron-rtd --registry-id 790709498068 --region  
us-west-2
```

```
kind: Deployment  
apiVersion: apps/v1  
metadata:  
  name: eks-neuron-test  
  labels:  
    app: eks-neuron-test  
    role: master  
spec:  
  replicas: 2  
  selector:  
    matchLabels:  
      app: eks-neuron-test  
      role: master  
  template:  
    metadata:  
      labels:  
        app: eks-neuron-test  
        role: master  
    spec:  
      containers:  
        - name: eks-neuron-test
```

```

image: 763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference-
neuron:1.15.4-neuron-py37-ubuntu18.04
command:
  - /usr/local/bin/entrypoint.sh
args:
  - --port=8500
  - --rest_api_port=9000
  - --model_name=resnet50_neuron
  - --model_base_path=s3://${your-bucket-of-models}/resnet50_neuron/
ports:
  - containerPort: 8500
  - containerPort: 9000
imagePullPolicy: IfNotPresent
env:
  - name: AWS_REGION
    value: "us-east-1"
  - name: S3_USE_HTTPS
    value: "1"
  - name: S3_VERIFY_SSL
    value: "0"
  - name: S3_ENDPOINT
    value: s3.us-east-1.amazonaws.com
  - name: AWS_LOG_LEVEL
    value: "3"
resources:
  limits:
    cpu: 4
    memory: 4Gi
    aws.amazon.com/neuron: 1
  requests:
    cpu: "1"
    memory: 1Gi
securityContext:
  capabilities:
    add:
      - IPC_LOCK

```

3. Men-deploy model.

```
kubectl apply -f rn50_deployment.yaml
```

4. Membuat file bernama rn50_service.yaml dengan konten berikut. Port HTTP dan gRPC dibuka untuk menerima permintaan prediksi.

```
kind: Service
apiVersion: v1
metadata:
  name: eks-neuron-test
  labels:
    app: eks-neuron-test
spec:
  type: ClusterIP
  ports:
    - name: http-tf-serving
      port: 8500
      targetPort: 8500
    - name: grpc-tf-serving
      port: 9000
      targetPort: 9000
  selector:
    app: eks-neuron-test
    role: master
```

5. Buat layanan Kubernetes untuk aplikasi Serving TensorFlow model Anda.

```
kubectl apply -f rn50_service.yaml
```

(Opsional) Buat prediksi terhadap layanan TensorFlow Serving Anda

1. Untuk menguji secara lokal, meneruskan port gRPC ke layanan eks-neuron-test.

```
kubectl port-forward service/eks-neuron-test 8500:8500 &
```

2. Membuat skrip Python yang disebut `tensorflow-model-server-infer.py` dengan konten berikut. Skrip ini menjalankan inferensi melalui gRPC, yang merupakan kerangka kerja layanan.

```
import numpy as np
import grpc
import tensorflow as tf
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.resnet50 import preprocess_input
from tensorflow_serving.apis import predict_pb2
from tensorflow_serving.apis import prediction_service_pb2_grpc
from tensorflow.keras.applications.resnet50 import decode_predictions
```

```

if __name__ == '__main__':
    channel = grpc.insecure_channel('localhost:8500')
    stub = prediction_service_pb2_grpc.PredictionServiceStub(channel)
    img_file = tf.keras.utils.get_file(
        './kitten_small.jpg',
        "https://raw.githubusercontent.com/aws-labs/mxnet-model-server/master/docs/
images/kitten_small.jpg")
    img = image.load_img(img_file, target_size=(224, 224))
    img_array = preprocess_input(image.img_to_array(img)[None, ...])
    request = predict_pb2.PredictRequest()
    request.model_spec.name = 'resnet50_inf1'
    request.inputs['input'].CopyFrom(
        tf.make_tensor_proto(img_array, shape=img_array.shape))
    result = stub.Predict(request)
    prediction = tf.make_ndarray(result.outputs['output'])
    print(decode_predictions(prediction))

```

3. Menjalankan penulisan untuk mengirimkan prediksi ke layanan Anda.

```
python3 tensorflow-model-server-infer.py
```

Contoh output adalah sebagai berikut.

```

[[('n02123045', 'tabby', 0.68817204), ('n02127052', 'lynx', 0.12701613),
 ('n02123159', 'tiger_cat', 0.08736559), ('n02124075', 'Egyptian_cat',
 0.063844085), ('n02128757', 'snow_leopard', 0.009240591)]]

```

Coba tutorial untuk menerapkan beban kerja Machine Learning di EKS

Jika Anda tertarik untuk menyiapkan platform dan kerangka kerja Machine Learning di EKS, jelajahi tutorial yang dijelaskan di halaman ini. Tutorial ini mencakup segala sesuatu mulai dari pola untuk memanfaatkan prosesor GPU sebaik-baiknya hingga memilih alat pemodelan hingga membangun kerangka kerja untuk industri khusus.

Bangun platform AI generatif di EKS

- [Terapkan Model AI Generatif di Amazon EKS](#)

- [Membangun JupyterHub Platform multi-penyewa di Amazon EKS](#)

Jalankan kerangka kerja AI generatif khusus di EKS

- [Percepat beban kerja pelatihan terdistribusi AI generatif Anda dengan NVIDIA NeMo Framework di Amazon EKS](#)
- [Berjalan TorchServe di Amazon Elastic Kubernetes Service](#)

Maksimalkan kinerja GPU NVIDIA untuk ML di EKS

- Terapkan berbagi GPU untuk menggunakan NVIDIA secara efisien GPUs untuk kluster EKS Anda:
[Berbagi GPU di Amazon EKS dengan pengiris waktu NVIDIA dan instans yang dipercepat EC2](#)
- Gunakan layanan mikro Multi-Instance GPUs (MIGs) dan NIM untuk menjalankan lebih banyak pod per GPU di kluster EKS Anda:
[Memaksimalkan pemanfaatan GPU dengan NVIDIA Multi-Instance GPU \(MIG\) di Amazon EKS: Menjalankan lebih banyak pod per GPU untuk meningkatkan kinerja](#)
- [Bangun dan terapkan sistem machine learning yang dapat diskalakan di Kubernetes dengan Kubeflow di AWS](#)

Jalankan beban kerja pengkodean video di EKS

- [Mengirimkan konten video dengan fraksional GPUs dalam wadah di Amazon EKS](#)

Mempercepat pemuatan gambar untuk beban kerja inferensi

- [Bagaimana H2O.ai mengoptimalkan dan mengamankan infrastruktur AI/MLnya dengan Karpenter dan Bottlerocket](#)

Memantau beban kerja ML

- [Memantau beban kerja GPU di Amazon EKS menggunakan layanan sumber terbuka AWS terkelola](#)
- [Aktifkan metrik GPU berbasis pod di Amazon CloudWatch](#)

Pengumuman untuk ML di EKS

- [Dukungan bottlerocket untuk NVIDIA GPUs](#)
- [Memanfaatkan GPU Multi-Instance NVIDIA \(MIG\) di Instans Amazon EC2 P4d di Amazon Elastic Kubernetes Service](#)
- [Menerapkan Instans P4d terkelola di Amazon Elastic Kubernetes Service dengan NVIDIA RDMA GPUDirect](#)

Memperluas kemampuan Amazon EKS dengan proyek open source

Proyek-proyek open-source ini memperluas fungsionalitas kluster Kubernetes yang berjalan di dalam atau di luar AWS, termasuk kluster yang dikelola oleh Amazon EKS.

Support untuk perangkat lunak yang digunakan untuk EKS

Saat meninjau dokumen Amazon EKS, Anda akan menemukan referensi ke berbagai alat dan perangkat lunak sumber terbuka di seluruh prosedur dan contoh kami. [Alat-alat ini termasuk Kubernetes Metrics Server dan Cert Manager.](#)

Harap dicatat bahwa perangkat lunak pihak ketiga atau sumber terbuka yang Anda pilih untuk digunakan berada di luar cakupan Perjanjian AWS Dukungan Anda. Manfaat menggunakan Kubernetes adalah komunitas open source yang aktif. Kami merekomendasikan bekerja secara langsung dengan komunitas sumber terbuka yang relevan dan pengelola proyek untuk membangun saluran dukungan yang sesuai untuk komponen tersebut. Untuk informasi selengkapnya, lihat [proyek lulus dan inkubasi](#) yang terkait dengan Cloud Native Computing Foundation (CNCF).

Ekosistem Kubernetes mencakup banyak proyek dan komponen yang datang dengan berbagai tingkat dukungan komunitas, waktu respons, dan kasus penggunaan yang dimaksudkan. Saat menerapkan teknologi ini bersama EKS, pastikan Anda memahami matriks dukungan untuk setiap komponen.

AWS mempertahankan komponen sumber terbuka yang kami integrasikan ke dalam bidang kontrol EKS. Ini termasuk saluran keamanan komprehensif kami yang mencakup verifikasi build, pemindaian kerentanan, pengujian validasi, dan manajemen tambalan untuk semua gambar kontainer dan binari yang kami distribusikan. Misalnya, bertanggung jawab AWS jawab atas [Kubernetes](#) API Server. Server API Kubernetes dilindungi oleh Perjanjian Tingkat [Layanan Amazon EKS](#). Anda dapat menggunakan [Paket Dukungan Amazon Web Services](#) untuk menyelesaikan masalah dengan server API Kubernetes, atau mendapatkan panduan umum.

Anda perlu meninjau dengan cermat dukungan yang ditawarkan untuk berbagai Add-on Amazon EKS. AWS add-on adalah satu-satunya jenis add-on Amazon EKS yang didukung penuh oleh AWS. AWS Add-on Marketplace terutama didukung oleh AWS Mitra. Add-on komunitas menerima dukungan siklus hidup dasar dari AWS. Untuk informasi selengkapnya, lihat [add-on Support](#).

Setiap add-on EKS, apa pun jenisnya, menerima dukungan siklus hidup dasar dari EKS termasuk add-on Marketplace. Dukungan siklus hidup dasar termasuk menginstal dan menghapus instalasi add-on. Untuk informasi selengkapnya tentang jenis Add-on Amazon EKS yang tersedia dan tingkat dukungan terkait, lihat [Lingkup Dukungan untuk add-on Amazon EKS](#). Untuk melihat add-on yang didukung sepenuhnya oleh AWS, lihat [add-on Amazon Web Services](#).

- Untuk informasi selengkapnya tentang praktik keamanan dan batasan dukungan kami, lihat [Keamanan di Amazon EKS](#).
- Untuk informasi selengkapnya tentang pengaya komunitas dan AWS marketplace yang tersedia melalui Add-on Amazon EKS, lihat Dukungan Pengaya [EKS](#).

Alat manajemen

Alat manajemen yang sesuai untuk kluster Amazon EKS dan Kubernetes.

eksctl

`eksctl` adalah alat CLI sederhana untuk membuat kluster di Amazon EKS.

- [URL Proyek](#)
- [Dokumentasi proyek](#)
- AWS blog open source: [eksctl: Amazon EKS cluster](#) dengan satu perintah

AWS pengendali untuk Kubernetes

Dengan AWS Controller untuk Kubernetes, Anda dapat membuat dan mengelola AWS sumber daya langsung dari kluster Kubernetes Anda.

- [URL Proyek](#)
- AWS blog open source: [operator AWS layanan untuk Kubernetes sekarang tersedia](#)

Flux CD

Flux adalah alat yang dapat Anda gunakan untuk mengelola konfigurasi kluster Anda dengan menggunakan Git. Hal tersebut menggunakan operator dalam kluster untuk memicu deployment dalam Kubernetes. Untuk informasi selengkapnya tentang operator, lihat [OperatorHub.io](#) di GitHub.

- [URL Proyek](#)
- [Dokumentasi proyek](#)

CDK untuk Kubernetes

Dengan CDK untuk Kubernetes (cdk8s), Anda dapat menentukan aplikasi dan komponen Kubernetes menggunakan bahasa pemrograman yang mudah dipahami. Aplikasi cdk8s dibentuk menjadi manifestes Kubernetes standar, yang dapat diterapkan ke kluster Kubernetes mana pun.

- [URL Proyek](#)
- [Dokumentasi proyek](#)
- AWS container blog: [Memperkenalkan cdk8s+](#): Didorong niat untuk objek Kubernetes APIs

Jaringan

Proyek jaringan yang berhubungan dengan kluster Amazon EKS dan Kubernetes.

Plugin Amazon VPC CNI untuk Kubernetes

Amazon EKS mendukung jaringan VPC asli melalui plugin Amazon VPC CNI untuk Kubernetes. Plugin memberikan alamat IP dari VPC Anda ke setiap Pod.

- [URL Proyek](#)
- [Dokumentasi proyek](#)

AWS Load Balancer Controller untuk Kubernetes

AWS Load Balancer Controller membantu mengelola AWS Elastic Load Balancer untuk kluster Kubernetes. Ini memenuhi sumber daya Kubernetes Ingress dengan menyediakan Application Load Balancers. AWS Ini memenuhi sumber daya layanan Kubernetes dengan menyediakan AWS Network Load Balancers.

- [URL Proyek](#)
- [Dokumentasi proyek](#)

ExternalDNS

ExternalDNS menyinkronkan layanan Kubernetes yang terekspos dan masuk ke penyedia DNS termasuk Amazon Route 53 dan Service Discovery. AWS

- [URL Proyek](#)
- [Dokumentasi proyek](#)

Machine learning

Machine Learning yang berkaitan dengan klaster Amazon EKS dan Kubernetes.

Kubeflow

Sebuah kit alat Machine Learning untuk Kubernetes.

- [URL Proyek](#)
- [Dokumentasi proyek](#)
- AWS blog sumber terbuka: [Kubeflow di Amazon EKS](#)

Auto Scaling

Proyek Auto Scaling yang berkaitan dengan klaster Amazon EKS dan Kubernetes.

Autoscaler klaster

Autoscaler klaster adalah alat yang secara otomatis menyesuaikan ukuran klaster Kubernetes berdasarkan CPU dan tekanan memori.

- [URL Proyek](#)
- [Dokumentasi proyek](#)
- Lokakarya Amazon EKS: [Cluster Autoscaler](#)

Karpenter

Karpenter adalah Kubernetes Node Autoscaler yang dibangun untuk fleksibilitas, kinerja, dan kesederhanaan.

- [URL Proyek](#)
- [Dokumentasi proyek](#)
- Lokakarya Amazon EKS: [Karpenter](#)

Eskalator

Eskalator adalah operasi batch atau tugas yang dioptimalkan dengan autoscaler horizontal untuk Kubernetes.

- [URL Proyek](#)
- [Dokumentasi proyek](#)

Pemantauan

Proyek pemantauan yang berkaitan dengan kluster Amazon EKS dan Kubernetes.

Prometheus

Prometheus adalah kit alat pemantauan dan peringatan sistem sumber terbuka.

- [URL Proyek](#)
- [Dokumentasi proyek](#)
- Lokakarya Amazon EKS: https://eksworkshop.com/intermediate/240_monitoring/

Integrasi berkelanjutan/deployment berkesinambungan

Proyek CI/CD yang berkaitan dengan kluster Amazon EKS dan Kubernetes.

Jenkins X

Solusi CI/CD untuk aplikasi cloud modern di kluster Amazon EKS dan Kubernetes.

- [URL Proyek](#)
- [Dokumentasi proyek](#)

Pelajari tentang fitur dan peta jalan baru Amazon EKS

Anda dapat mempelajari tentang fitur Amazon EKS baru dengan menggulir ke umpan Apa yang Baru di halaman [Apa yang Baru dengan AWS](#). Anda juga dapat meninjau [peta jalan](#) GitHub, yang memungkinkan Anda mengetahui tentang fitur dan prioritas yang akan datang sehingga Anda dapat merencanakan bagaimana Anda ingin menggunakan Amazon EKS di masa depan. Anda dapat memberikan umpan balik langsung kepada kami tentang prioritas panduan (roadmap).

Riwayat dokumen

Tabel berikut menjelaskan beberapa pembaruan utama dan fitur baru untuk Panduan Pengguna Amazon EKS. Untuk menerima pemberitahuan saat tabel ini mendapat entri baru, Anda dapat berlangganan URL berikut dengan pembaca RSS:

<https://docs.aws.amazon.com/eks/latest/userguide/doc-history.rss>

Perubahan	Deskripsi	Tanggal
Pembaruan konten pemecahan masalah VPC CNI	Memperluas halaman pemecahan masalah untuk kebijakan jaringan Kubernetes di CNI VPC. Menambahkan izin CRDs dan RBAC, dan 12 masalah yang diketahui.	Juni 30, 2025
Pembaruan versi platform Amazon EKS	Ini adalah versi platform baru dengan perbaikan dan peningkatan keamanan. Tidak ada versi patch baru Kubernetes di versi platform ini.	Juni 26, 2025
AWS pembaruan kebijakan terkelola	Menambahkan <code>ssmmessages:OpenDataChannel</code> izin ke <code>AmazonEKSLocalOutpostServiceRolePolicy</code> .	Juni 26, 2025
AWS pembaruan kebijakan terkelola	Menambahkan izin <code>AmazonEKSComputePolicy</code> ke <code>AmazonEKSServiceRolePolicy</code> dan mengizinkan Mode Otomatis Amazon EKS meluncurkan instans dengan menggunak	Juni 20, 2025

an Reservasi Kapasitas EC2 Sesuai Permintaan di akun Anda. Selain itu, tambahkan izin ke Amazon EKS AmazonEKSCoordinatePolicy untuk membuat peran terkait layanan EC2 Spot atas nama Anda.

[Pembaruan kebijakan terkelola](#)

Ditambahkan AmazonEKS DashboardConsoleReadOnly kebijakan.

Juni 19, 2025

[Pembaruan Mode Otomatis Amazon EKS ke NodeClass](#)

NodeClass Template untuk node Mode Otomatis menambahkan konfigurasi untuk subnet pod terpisah. Ini menambahkan kunci opsional podSubnetSelectorTerms dan podSecurityGroupSelectorTerms untuk mengatur subnet dan grup keamanan untuk pod.

Juni 13, 2025

[Targetkan peran sekunder dan lintas akun dengan Identitas Pod EKS](#)

Amazon EKS menambahkan peran IAM target ke EKS Pod Identities untuk rantai peran otomatis. Anda dapat menggunakan ini untuk secara otomatis mengambil peran di akun lain dan EKS Pod Identity memutar kredensya l sementara. Setiap asosiasi Pod Identity harus memiliki peran IAM dalam akun yang sama untuk diasumsikan terlebih dahulu, kemudian menggunakan peran tersebut untuk mengambil peran target.

Juni 11, 2025

[Pembaruan versi platform Amazon EKS](#)

Ini adalah versi platform baru dengan perbaikan dan peningkatan keamanan. Tidak ada versi patch baru Kubernetes di versi platform ini.

Juni 11, 2025

[Ekspansi AWS Wilayah Amazon EKS](#)

Amazon EKS sekarang tersedia di AWS Wilayah Asia Pasifik (Taipei) (ap-east-2).

Juni 6, 2025

[IPv6kontrol akses untuk titik akhir publik dual-stack untuk cluster baru IPv6](#)

Juni 5, 2025

Pembaruan versi platform Amazon EKS	Ini adalah versi platform baru dengan perbaikan dan peningkatan keamanan. Ini termasuk versi patch baru dari Kubernetes1.32.5,, 1.31.9 dan. 1.30.13	30 Mei 2025
Wawasan cluster baru untuk EKS Hybrid Nodes	Amazon EKS menambahkan wawasan cluster baru yang memeriksa konfigurasi node hybrid Anda. Pemeriksaan wawasan ini akan memperingatkan Anda tentang masalah dengan node dan pod lokal serta konfigurasi jaringan jarak jauh klaster.	29 Mei 2025
Versi Kubernetes 1.33	Menambahkan 1.33 dukungan versi Kubernetes untuk klaster baru dan peningkatan versi.	29 Mei 2025
Dukungan add-on untuk driver Amazon FSx CSI	Sekarang Anda dapat menggunakan AWS Management Console, AWS CLI, dan API untuk mengelola driver Amazon FSx CSI.	Mei 23, 2025
Edit pencakar Prometheus di konsol	Anda sekarang dapat mengedit Layanan Terkelola Amazon untuk pencakar Prometheus di konsol Amazon EKS.	22 Mei 2025
Pembaruan kebijakan terkelola	Ditambahkan AmazonEKS DashboardServiceRolePolicy kebijakan.	21 Mei 2025

Pembaruan versi platform Amazon EKS	Ini adalah versi platform baru dengan perbaikan dan peningkatan keamanan. Tidak ada versi patch baru Kubernetes di versi platform ini.	16 Mei 2025
Halaman baru untuk Amazon FSx untuk kinerja Lustre	Menambahkan topik baru dengan detail tentang mengoptimalkan Amazon FSx untuk kinerja Lustre.	2 Mei 2025
Pembaruan Mode Otomatis Amazon EKS ke NodeClass	NodeClass Template untuk node Mode Otomatis menambahkan konfigurasi untuk proxy jaringan maju. Ini menambahkan kunci opsional advancedNetworking untuk mengatur proxy HTTPS Anda.	April 30, 2025
Bottlerocket untuk node hibrida	Bottlerocket sekarang tersedia untuk EKS Hybrid Nodes.	April 29, 2025
Pembaruan versi platform Amazon EKS	Ini adalah versi platform baru dengan perbaikan dan peningkatan keamanan. Tidak ada versi patch baru Kubernetes di versi platform ini.	April 29, 2025
Halaman konsep baru untuk jaringan hybrid	Ditambahkan halaman untuk konsep EKS Hybrid Nodes. Ini mencakup jaringan lokal dan cloud secara detail dengan diagram.	April 18, 2025

[AWS pembaruan kebijakan terkelola](#)

Menambahkan izin `AmazonEKSClusterPolicy` untuk mengizinkan Amazon EKS ke antarmuka jaringan elastis yang dibuat oleh VPC CNI. Ini diperlukan agar EKS dapat membersihkan antarmuka jaringan elastis yang tertinggal jika VPC CNI berhenti secara tak terduga.

April 16, 2025

[AWS pembaruan kebijakan terkelola](#)

Menambahkan izin `AmazonEKSServiceRolePolicy` untuk mengizinkan AI/ML pelanggan EKS menambahkan aturan Egress ke grup keamanan EKS Cluster default.

April 14, 2025

[Kesehatan node untuk EKS Hybrid Nodes](#)

Anda dapat menggunakan `eks-node-monitoring-agent` pada node hybrid, mulai dari versi `1.2.0-eksbuild.1`. Jalankan `eks-node-monitoring-agent` sebagai add-on Amazon EKS untuk mendeteksi dan menunjukkan masalah kesehatan.

Maret 31, 2025

[EKS Hybrid Nodes untuk cluster yang ada](#)

Anda sekarang dapat menambahkan, mengubah, atau menghapus konfigurasi node hibrida dari cluster yang ada. Sebelumnya, Anda hanya bisa menambahkan konfigurasi node hybrid ke cluster baru saat Anda membuatnya. Dengan Amazon EKS Hybrid Nodes, Anda dapat menggunakan infrastruktur lokal dan edge sebagai node di kluster Amazon EKS. AWS mengelola bidang kontrol Kubernetes yang AWS di-host dari kluster Amazon EKS, dan Anda mengelola node hibrida yang berjalan di lingkungan lokal atau edge Anda.

Maret 31, 2025

[Rollback: Mencegah peningkatan yang tidak disengaja dengan wawasan cluster](#)

Amazon EKS telah mengembalikan sementara fitur yang mengharuskan Anda menggunakan `--force` flag untuk memutakhirkan kluster Anda saat ada masalah wawasan kluster tertentu. Untuk informasi selengkapnya, lihat [Pengembalian sementara untuk menerapkan wawasan pemutakhiran pada versi kluster pembaruan pada](#) GitHub

Maret 28, 2025

Bottlerocket FIPS AMIs	Bottlerocket FIPS sekarang tersedia dalam grup AMIs node terkelola standar.	Maret 27, 2025
AWS pembaruan kebijakan terkelola	Menambahkan izin AmazonEKSServiceRolePolicy untuk mengizinkan Amazon EKS menghentikan EC2 instans yang dibuat oleh Mode Otomatis. Menambahkan izin AmazonEKSServiceRolePolicy untuk mengizinkan Amazon EKS menghentikan EC2 instans yang dibuat oleh Mode Otomatis.	Februari 28, 2025
Memperbarui strategi untuk grup node terkelola	Anda sekarang dapat menggunakan strategi pembaruan untuk mengonfigurasi proses pembaruan versi untuk grup node terkelola. Ini memperkenalkan strategi pembaruan minimal untuk menghentikan node sebelum membuat yang baru, yang berguna dalam lingkungan terbatas kapasitas. Strategi pembaruan default melanjutkan perilaku yang ada.	Januari 27, 2025
Versi Kubernetes 1.32	Menambahkan 1.32 dukungan versi Kubernetes untuk kluster baru dan peningkatan versi.	Januari 23, 2025

[Ekspansi AWS Wilayah
Amazon EKS](#)

Amazon EKS sekarang tersedia di Wilayah Asia Pasifik (Thailand) (ap-southeast-7) dan Wilayah Meksiko (Tengah) mx-central-1 (AWS). Mode Otomatis EKS dan Titik Akhir VPC untuk EKS API tidak tersedia di kedua Wilayah.

Januari 14, 2025

[AWS pembaruan kebijakan
terkelola](#)

Menambahkan beberapa izin AmazonEBSCSIDriver Policy untuk memungkinkan Driver Amazon EBS CSI memulihkan semua snapshot, mengaktifkan Pemulihan Snapshot Cepat (FSR) pada volume EBS, dan memodifikasi tag pada volume.

Januari 13, 2025

[AWS pembaruan kebijakan
terkelola](#)

Menambahkan izin keAmazonEKSLoadBalancingPolicy .

Desember 26, 2024

[Wawasan cluster yang
diperbarui](#)

Wawasan peningkatan Amazon EKS sekarang akan memperingatkan tentang lebih banyak masalah kesehatan klaster dan kompatibilitas versi. Ini dapat mendeteksi masalah antara komponen Kubernetes dan Amazon EKS yang berbeda seperti kubelet,, kube-proxy dan add-on Amazon EKS.

Desember 20, 2024

[Agen pemantauan node dan perbaikan otomatis](#)

Anda dapat menggunakan yang baru eks-node-monitoring-agent sebagai add-on Amazon EKS untuk mendeteksi dan menunjukkan masalah kesehatan. Anda juga dapat mengaktifkan perbaikan otomatis node untuk secara otomatis mengganti node ketika masalah terdeteksi.

Desember 16, 2024

[Node Hibrida Amazon EKS](#)

Sekarang Anda dapat menjalankan node lokal yang terhubung ke kluster Amazon EKS. Dengan Amazon EKS Hybrid Nodes, Anda dapat menggunakan infrastruktur lokal dan edge sebagai node di kluster Amazon EKS. AWS mengelola bidang kontrol Kubernetes yang AWS di-host dari kluster Amazon EKS, dan Anda mengelola node hibrida yang berjalan di lingkungan lokal atau edge Anda.

Desember 1, 2024

Mode Otomatis Amazon EKS	Amazon EKS Auto Mode sepenuhnya mengotomatiskan manajemen infrastruktur kluster Kubernetes untuk komputasi, penyimpanan, dan jaringan aktif. AWS Ini menyederhanakan pengelolaan Kubernetes dengan menyediakan infrastruktur secara otomatis, memilih instans komputasi yang optimal, menskalakan sumber daya secara dinamis, terus mengoptimalkan biaya, menambal sistem operasi, dan mengintegrasikan dengan layanan keamanan. AWS	Desember 1, 2024
Pembaruan versi platform Amazon EKS	Ini adalah versi platform baru dengan perbaikan dan peningkatan keamanan. Ini termasuk versi patch baru dari Kubernetes 1.31.2, 1.30.6, 1.29.10 dan 1.28.15	November 22, 2024
AWS pembaruan kebijakan terkelola	Diperbarui AWSServiceRoleForAmazonEKStoodegroup untuk kompatibilitas dengan wilayah Tiongkok.	November 22, 2024
Kubernetes versi 1.30 sekarang tersedia untuk cluster lokal di Outposts AWS	Anda sekarang dapat membuat cluster lokal Amazon EKS di AWS Outposts menggunakan Kubernetes versi 1.30.	November 21, 2024

[AWS pembaruan kebijakan terkelola](#)

EKS memperbarui kebijakan AWS terkelolaAmazonEKS LocalOutpostClusterPolicy . Menambahkan ec2:DescribeAvailabilityZones izin agar AWS Cloud Controller Manager pada bidang kontrol cluster dapat mengidentifikasi Availability Zone tempat setiap node berada.

November 21, 2024

[Bottlerocket AMIs yang menggunakan FIPS 140-3](#)

Bottlerocket tersedia yang AMIs telah dikonfigurasi sebelumnya untuk menggunakan modul kriptografi tervalidasi FIPS 140-3. Ini termasuk Modul Kriptografi Amazon Linux 2023 Kernel Crypto API dan Modul Kriptografi AWS-LC.

November 20, 2024

[AWS pembaruan kebijakan terkelola](#)

AWSServiceRoleForAmazonEKSNodegroup Kebijakan yang diperbarui ec2:RebootInstances untuk mengizinkan instance yang dibuat oleh grup node terkelola Amazon EKS. Membatasi ec2:CreateTags izin untuk EC2 sumber daya Amazon.

November 20, 2024

Dasbor observabilitas	Dasbor observabilitas membantu Anda mendeteksi, memecahkan masalah, dan memperbaiki masalah dengan cepat. Ada juga metrik CloudWatch vended baru yang tersedia di namespace. AWS/EKS	November 18, 2024
AWS pembaruan kebijakan terkelola	EKS memperbarui kebijakan AWS terkelolaAmazonEKSServiceRolePolicy . Menambahkan izin untuk kebijakan akses EKS, manajemen penyeimbang beban, dan pembersihan sumber daya kluster otomatis.	November 16, 2024
Pembuatan peran baru di konsol untuk add-on yang mendukung EKS Pod Identity	Ada langkah-langkah baru saat menggunakan konsol untuk membuat atau memperbarui add-on yang mendukung Identitas Pod EKS di mana Anda dapat secara otomatis membuat peran IAM dengan nama, kebijakan peran, dan kebijakan kepercayaan yang sesuai untuk add-on tersebut.	November 15, 2024
Grup node terkelola di AWS Local Zones	Grup node terkelola sekarang dapat dibuat di AWS Local Zones.	November 15, 2024
Metrik baru tersedia	Ada metrik baru yang tersedia di bawah grup <code>metrics.eks.amazonaws.com</code> API.	November 11, 2024

AWS pembaruan kebijakan terkelola	EKS memperbarui kebijakan AWS terkelolaAmazonEKSComputePolicy . Izin sumber daya yang diperbarui untuk iam:AddRoleToInstanceProfile tindakan tersebut.	November 7, 2024
AWS pembaruan kebijakan terkelola	EKS menambahkan kebijakan AWS terkelola baru: AmazonEKSComputePolicy	November 1, 2024
AWS pembaruan kebijakan terkelola	Menambahkan izin keAmazonEKSClusterPolicy . Menambahkan ec2:DescribeInstanceTopology izin untuk mengizinkan Amazon EKS melampirkan informasi topologi ke node sebagai label.	November 1, 2024
AWS pembaruan kebijakan terkelola	EKS menambahkan kebijakan AWS terkelola baru: AmazonEKSBlockStoragePolicy	Oktober 30, 2024
AWS pembaruan kebijakan terkelola	EKS menambahkan kebijakan AWS terkelola baru: AmazonEKSLoadBalancingPolicy	Oktober 30, 2024

[AWS pembaruan kebijakan terkelola](#)

Menambahkan cloudwatch:PutMetricData izin AmazonEKSServiceRolePolicy untuk mengizinkan Amazon EKS mempublikasikan metrik ke Amazon CloudWatch

Oktober 29, 2024

[AWS pembaruan kebijakan terkelola](#)

EKS menambahkan kebijakan AWS terkelola baru: AmazonEKSTerminatingPolicy

Oktober 28, 2024

[Titik akhir tumpukan ganda untuk cluster baru IPv6](#)

Connect ke IPv6 cluster baru dengan eksctl. `eksctl create cluster --region region --api --endpoint region.api.aws` endpoint yang dual-stack. Titik akhir ini dikembalikan saat Anda menjelaskan kluster ini. kubectl dan klien API Kubernetes lainnya diIPv4,IPv6, atau lingkungan dual-stack dapat menyelesaikan dan terhubung ke titik akhir ini untuk kluster publik atau pribadi.

Oktober 21, 2024

[AWS pembaruan kebijakan terkelola](#)

Menambahkan `autoscaling:ResumeProcesses` dan `autoscaling:SuspendProcesses` ke wilayah `AWSServiceRoleForAmazonEKS` di Tiongkok untuk diintegrasikan dengan Pengontrol Pemulihan Aplikasi Amazon untuk EKS. Tidak ada perubahan ke wilayah lain.

Oktober 21, 2024

[AL2023 dipercepat AMIs](#)

Anda sekarang dapat menggunakan instance akselerasi NVIDIA dan AWS Neuron AMIs berdasarkan AL2023.

Oktober 11, 2024

[Format sumber baru](#)

Kami telah beralih ke format sumber baru dengan beberapa perubahan tata letak. Ada masalah pemformatan kecil sementara yang sedang kami tangani.

Oktober 10, 2024

[AWS pembaruan kebijakan terkelola](#)

Menambahkan izin ke `AmazonEKSServicePolicy` dan `AmazonEKSServiceRolePolicy`. Izin tag yang ditambahkan `ec2:GetSecurityGroupsForVpc` dan terkait untuk memungkinkan EKS membaca informasi grup keamanan dan memperbarui tag terkait.

Oktober 10, 2024

AWS pembaruan kebijakan terkelola - Kebijakan baru	EKS menambahkan kebijakan AWS terkelola baru.	Oktober 3, 2024
Versi Kubernetes 1.31	Menambahkan 1.31 dukungan versi Kubernetes untuk klaster baru dan peningkatan versi.	September 24, 2024
AWS pembaruan kebijakan terkelola - Pembaruan ke kebijakan yang ada	Amazon EKS memperbarui kebijakan AWS terkelola yang ada.	Agustus 21, 2024
Kubernetes versi 1.29 sekarang tersedia untuk cluster lokal di Outposts AWS	Anda sekarang dapat membuat cluster lokal Amazon EKS di AWS Outposts menggunakan Kubernetes versi 1.29.	Agustus 20, 2024
Identitas Pod AWS GovCloud EKS di (AS)	Amazon EKS Pod Identities mengaitkan peran IAM dengan akun layanan Kubernetes. Dengan fitur ini, Anda tidak perlu lagi memberikan izin tambahan ke peran IAM node. Dengan cara ini, Pod pada node itu dapat memanggil AWS APIs. Tidak seperti peran IAM untuk akun layanan, EKS Pod Identities sepenuhnya berada di dalam EKS; Anda tidak memerlukan penyedia identitas OIDC.	Agustus 14, 2024
Pembaruan konten berbasis skenario	Kami mengganti nama dan memperbarui topik agar lebih didorong oleh skenario di seluruh panduan.	Agustus 9, 2024

[Titik akhir antarmuka VPC tumpukan ganda untuk Amazon EKS](#)

Anda sekarang dapat membuat titik akhir antarmuka VPC dual-stack untuk Amazon EKS dengan keduanya IPv4 dan IPv6 alamat IP dan nama DNS.

Agustus 7, 2024

[Titik akhir dual-stack baru untuk Amazon EKS dengan alamat APIs IPv6](#)

EKS API untuk membuat dan mengelola cluster, dan penerbit OIDC URLs untuk cluster memiliki titik akhir dual-stack baru. Nama DNS baru untuk Amazon EKS API adalah `eks.region.api.aws` yang menyelesaikan alamat dan IPv4 alamat. IPv6 Cluster baru memiliki URL penerbit OIDC dual-stack baru (`oidc-eks.region.api.aws`).

Agustus 1, 2024

[Blok Kapasitas untuk grup node terkelola](#)

Anda sekarang dapat menggunakan Blok Kapasitas untuk grup node terkelola.

Juli 1, 2024

[Koleksi metrik Grup Auto Scaling diaktifkan secara default](#)

Grup node terkelola Amazon EKS sekarang mengaktifkan metrik grup EC2 Auto Scaling Amazon secara default tanpa biaya tambahan. Sebelumnya, Anda harus melakukan beberapa langkah untuk mengaktifkan fitur ini.

Juni 28, 2024

AWS pembaruan kebijakan terkelola - Pembaruan ke kebijakan yang ada	Amazon EKS memperbarui kebijakan AWS terkelola yang ada.	27 Juni 2024
Versi Kubernetes 1.26	Versi Kubernetes sekarang dalam 1.26 dukungan yang diperluas.	Juni 12, 2024
Perbaikan referensi informasi AMI	Kami melakukan perbaikan pada referensi informasi AMI, khususnya untuk Bottlerocket.	Juni 12, 2024
Versi Kubernetes 1.30	Menambahkan 1.30 dukungan versi Kubernetes untuk klaster baru dan peningkatan versi.	23 Mei 2024
Penskalaan Otomatis CoreDNS	Autoscaler CoreDNS akan secara dinamis mengadaptasi jumlah replika penyebaran CoreDNS di cluster EKS berdasarkan jumlah node dan inti CPU. Fitur ini berfungsi untuk v1.9 CoreDNS dan versi platform terbaru dari versi rilis 1.25 EKS dan yang lebih baru.	14 Mei 2024
Pembaruan versi platform Amazon EKS	Ini adalah versi platform baru dengan perbaikan dan peningkatan keamanan. Ini termasuk versi patch baru dari Kubernetes 1.29.4,, 1.28.9 dan 1.27.13	14 Mei 2024

CloudWatch Dukungan Container Insights untuk Windows	Add-on Amazon CloudWatch Observability Operator sekarang juga memungkinkan Container Insights pada node pekerja Windows di cluster.	April 10, 2024
Konsep Kubernetes	Menambahkan topik konsep Kubernetes baru.	April 5, 2024
Merestrukturisasi Akses dan Konten IAM	Pindahkan halaman yang ada terkait dengan topik akses dan IAM, seperti peta konfigurasi auth, entri akses, ID Pod, dan IRSA ke bagian baru. Merevisi isi ikhtisar.	April 2, 2024
Dukungan Bottlerocket OS untuk driver Amazon S3 CSI	Driver Mountpoint untuk Amazon S3 CSI sekarang kompatibel dengan Bottlerocket.	Maret 13, 2024
AWS pembaruan kebijakan terkelola - Pembaruan ke kebijakan yang ada	Amazon EKS memperbarui kebijakan AWS terkelola yang ada.	Maret 4, 2024
Amazon Linux 2023	Amazon Linux 2023 (AL2023) adalah sistem operasi berbasis Linux baru yang dirancang untuk menyediakan lingkungan yang aman, stabil, dan berkinerja tinggi untuk aplikasi cloud Anda.	Februari 29, 2024

[EKS Pod Identity dan IRSA mendukung sidecar di Kubernetes 1.29](#)

Di Kubernetes 1.29, kontainer sespan tersedia di cluster Amazon EKS. Kontainer sidecar didukung dengan peran IAM untuk akun layanan atau EKS Pod Identity. Untuk informasi selengkapnya tentang sidecar, lihat [Sidecar Container di dokumentasi Kubernetes](#).

Februari 26, 2024

[Versi Kubernetes 1.29](#)

Menambahkan 1.29 dukungan versi Kubernetes untuk kluster baru dan peningkatan versi.

23 Januari 2024

[Rilis lengkap: Amazon EKS Extended Support untuk versi Kubernetes](#)

Dukungan versi Kubernetes yang diperluas memungkinkan Anda untuk tinggal di versi Kubernetes tertentu selama lebih dari 14 bulan.

Januari 16, 2024

[Deteksi kesehatan kluster
Amazon EKS di AWS Cloud](#)

Amazon EKS mendeteksi masalah dengan kluster Amazon EKS Anda dan infrastruktur prasyarat kluster dalam kesehatan kluster. Anda dapat melihat masalah dengan kluster EKS Anda di AWS Management Console dan di health cluster di EKS API. Masalah ini merupakan tambahan dari masalah yang terdeteksi oleh dan ditampilkan oleh konsol. Sebelumnya, kesehatan cluster hanya tersedia untuk cluster lokal di AWS Outposts.

28 Desember 2023

[Wawasan cluster](#)

Anda sekarang bisa mendapatkan rekomendasi pada cluster Anda berdasarkan pemeriksaan berulang.

20 Desember 2023

[Ekspansi AWS Wilayah
Amazon EKS](#)

Amazon EKS sekarang tersedia di Wilayah Kanada Barat (Calgary) ca-west-1 (AWS).

20 Desember 2023

[Sekarang Anda dapat memberikan peran IAM dan akses pengguna ke kluster Anda menggunakan entri akses](#)

Sebelum pengenalan entri akses, Anda memberikannya peran IAM dan akses pengguna ke kluster Anda dengan menambahkan entri ke `aws-auth ConfigMap`. Sekarang setiap cluster memiliki mode akses, dan Anda dapat beralih menggunakan entri akses pada jadwal Anda. Setelah Anda beralih mode, Anda dapat menambahkan pengguna dengan menambahkan entri akses di AWS CLI, AWS CloudFormation, dan AWS SDKs.

18 Desember 2023

[Pembaruan versi platform Amazon EKS](#)

Ini adalah versi platform baru dengan perbaikan dan peningkatan keamanan. Ini termasuk versi patch baru dari Kubernetes 1.28.4, 1.27.8, 1.26.11 dan 1.25.16.

Desember 12, 2023

[Mountpoint untuk driver Amazon S3 CSI](#)

Anda sekarang dapat menginstal driver Mountpoint untuk Amazon S3 CSI di cluster Amazon EKS.

27 November 2023

[Aktifkan metrik Prometheus saat membuat cluster](#)

Di AWS Management Console, Anda sekarang dapat mengaktifkan metrik Prometheus saat membuat cluster. Anda juga dapat melihat detail scraper Prometheus di tab Observability.

26 November 2023

[Identitas Amazon EKS Pod](#)

Amazon EKS Pod Identities mengaitkan peran IAM dengan akun layanan Kubernetes. Dengan fitur ini, Anda tidak perlu lagi memberikan izin tambahan ke peran IAM node. Dengan cara ini, Pod pada node itu dapat memanggil AWS APIs. Tidak seperti peran IAM untuk akun layanan, EKS Pod Identities sepenuhnya berada di dalam EKS; Anda tidak memerlukan penyedia identitas OIDC.

26 November 2023

[AWS pembaruan kebijakan terkelola - Pembaruan ke kebijakan yang ada](#)

Amazon EKS memperbarui kebijakan AWS terkelola yang ada.

26 November 2023

[Pengontrol snapshot CSI](#)

Anda sekarang dapat menginstal pengontrol snapshot CSI untuk digunakan dengan driver CSI yang kompatibel, seperti driver Amazon EBS CSI.

17 November 2023

[Penulisan ulang topik Operator ADOT](#)

Dukungan add-on Amazon EKS untuk bagian Operator ADOT berlebihan dengan Distro untuk AWS dokumentasi. OpenTelemetry Kami memigrasikan informasi penting yang tersisa ke sumber daya itu untuk mengurangi informasi yang ketinggalan zaman dan tidak konsisten.

14 November 2023

[Dukungan add-on CoreDNS EKS untuk metrik Prometheus](#)

v1.8.7-eksbuild.8 Versi v1.10.1-eksbuild.5 , v1.9.3-eksbuild.9 , dan dari add-on EKS untuk CoreDNS mengekspos port yang CoreDNS menerbitkan metrik, dalam layanan kube-dns Ini memudahkan untuk menyertakan metrik CoreDNS dalam sistem pemantauan Anda.

10 November 2023

[Pengaya Operator CloudWatch Observabilitas Amazon EKS](#)

Menambahkan halaman Operator CloudWatch Observabilitas Amazon EKS.

6 November 2023

[Blok Kapasitas untuk instans P5 yang dikelola sendiri di AS Timur \(Ohio\)](#)

Di US East (Ohio), Anda sekarang dapat menggunakan Blok Kapasitas untuk instans P5 yang dikelola sendiri.

31 Oktober 2023

[Cluster mendukung modifikasi subnet dan grup keamanan](#)

Anda dapat memperbarui cluster untuk mengubah subnet dan grup keamanan mana yang digunakan cluster. Anda dapat memperbarui dari AWS Management Console, versi terbaru dari AWS CLI, AWS CloudFormation, dan eksctl versi `v0.164.0-rc.0` atau yang lebih baru. Anda mungkin perlu melakukan ini untuk menyediakan subnet dengan lebih banyak alamat IP yang tersedia untuk berhasil meningkatkan versi cluster.

24 Oktober 2023

[Peran cluster dan peran grup node terkelola mendukung kebijakan AWS Identity and Access Management yang dikelola pelanggan](#)

Anda dapat menggunakan kebijakan IAM khusus pada peran klaster, alih-alih kebijakan AWS terkelola [EKSClusterKebijakan Amazon](#). Selain itu, Anda dapat menggunakan kebijakan IAM khusus pada peran node dalam grup node terkelola, bukan kebijakan EKSWorker NodePolicy AWS terkelola [Amazon](#). Lakukan ini untuk membuat kebijakan dengan hak istimewa paling sedikit untuk memenuhi persyaratan kepatuhan yang ketat.

23 Oktober 2023

Perbaiki tautan ke instalasi eksctl	Perbaiki link install untuk eksctl setelah halaman dipindahkan.	Oktober 6, 2023
Rilis pratinjau: Amazon EKS Extended Support untuk versi Kubernetes	Dukungan versi Kubernetes yang diperluas memungkinkan Anda untuk tinggal di versi Kubernetes tertentu selama lebih dari 14 bulan.	4 Oktober 2023
Hapus referensi ke integrasi AWS App Mesh	Integrasi Amazon EKS dengan AWS App Mesh tetap hanya untuk pelanggan App Mesh yang sudah ada.	September 29, 2023
Versi Kubernetes 1.28	Menambahkan 1.28 dukungan versi Kubernetes untuk klaster baru dan peningkatan versi.	26 September 2023
Cluster yang ada mendukung penegakan kebijakan jaringan Kubernetes di plugin Amazon VPC CNI untuk Kubernetes	Anda dapat menggunakan kebijakan jaringan Kubernetes di klaster yang ada dengan plugin Amazon VPC CNI untuk Kubernetes, alih-alih memerlukan solusi pihak ketiga. Anda dapat menggunakan kebijakan jaringan Kubernetes di klaster yang ada dengan plugin Amazon VPC CNI untuk Kubernetes, alih-alih memerlukan solusi pihak ketiga.	15 September 2023

[CoreDNS Amazon EKS add-on mendukung modifikasi PDB](#)

Anda dapat memodifikasi add-on EKS untuk CoreDNS dalam v1.9.3-eksbuild.7 versi dan yang lebih baru dan yang lebih baru. PodDisruptionBudget v1.10.1-eksbuild.4

15 September 2023

[Dukungan Amazon EKS untuk subnet bersama](#)

[Persyaratan dan pertimbangan subnet Bersama Baru untuk membuat kluster](#) Amazon EKS di subnet bersama.

7 September 2023

[Pembaruan untuk Apa itu Amazon EKS?](#)

Menambahkan [kasus penggunaan umum](#) baru dan topik [Arsitektur](#). Segarkan topik lainnya.

September 6, 2023

[Penegakan kebijakan jaringan Kubernetes di plugin Amazon VPC CNI untuk Kubernetes](#)

Anda dapat menggunakan kebijakan jaringan Kubernetes dengan plugin Amazon VPC CNI untuk Kubernetes, alih-alih memerlukan solusi pihak ketiga. Anda dapat menggunakan kebijakan jaringan Kubernetes dengan plugin Amazon VPC CNI untuk Kubernetes, alih-alih memerlukan solusi pihak ketiga.

29 Agustus 2023

[Ekspansi AWS Wilayah Amazon EKS](#)

Amazon EKS sekarang tersedia di AWS Wilayah Israel (Tel Aviv) (il-central-1).

1 Agustus 2023

Penyimpanan fana yang dapat dikonfigurasi untuk Fargate	Anda dapat meningkatkan jumlah total penyimpanan sementara untuk setiap Pod yang berjalan di Amazon EKS Fargate.	31 Juli 2023
Dukungan add-on untuk driver Amazon EFS CSI	Anda sekarang dapat menggunakan AWS Management Console, AWS CLI, dan API untuk mengelola driver Amazon EFS CSI.	26 Juli 2023
AWS pembaruan kebijakan terkelola - Kebijakan baru	Amazon EKS menambahkan kebijakan AWS terkelola baru.	26 Juli 2023
Pembaruan versi Kubernetes untuk 1.27, 1.26, 1.25, dan 1.24 sekarang tersedia untuk cluster lokal di Outposts AWS	Pembaruan versi Kubernetes ke 1.27.3, 1.26.6, 1.25.11, dan 1.24.15 sekarang tersedia untuk cluster lokal di Outposts AWS	Juli 20, 2023
Dukungan awalan IP untuk node Windows	Menetapkan prefiks IP ke node Anda dapat memungkinkan Anda untuk meng-host jumlah Pod yang jauh lebih tinggi pada node Anda daripada yang dapat Anda lakukan saat menetapkan alamat IP sekunder individual ke node Anda.	6 Juli 2023
Amazon FSx untuk driver OpenZFS CSI	Anda sekarang dapat menginstal driver Amazon FSx untuk OpenZFS CSI di kluster Amazon EKS.	Juni 30, 2023

Pod pada node Linux dalam IPv4 cluster sekarang dapat berkomunikasi dengan IPv6 endpoint.	Setelah menetapkan IPv6 alamat ke node Anda, alamat Pods Anda adalah IPv4 alamat jaringan yang diterjemahkan ke IPv6 alamat node yang sedang berjalan.	19 Juni 2023
Grup simpul terkelola Windows di Wilayah AWS GovCloud (AS)	Di Wilayah AWS GovCloud (AS), grup node terkelola Amazon EKS sekarang dapat menjalankan wadah Windows.	30 Mei 2023
Versi Kubernetes 1.27	Menambahkan 1.27 dukungan versi Kubernetes untuk kluster baru dan peningkatan versi.	24 Mei 2023
Versi Kubernetes 1.26	Menambahkan 1.26 dukungan versi Kubernetes untuk kluster baru dan peningkatan versi.	11 April 2023
GMSA Tanpa Domain	Anda sekarang dapat menggunakan GMSA tanpa domain dengan Pod Windows.	Maret 27, 2023
Ekspansi AWS Wilayah Amazon EKS	Amazon EKS sekarang tersedia di AWS Wilayah Asia Pasifik (Melbourneap-southeast-4) ().	10 Maret 2023
Driver CSI Cache File Amazon	Anda sekarang dapat menginstal driver Cache File CSI Amazon di kluster Amazon EKS.	3 Maret 2023

Kubernetes versi 1.25 sekarang tersedia untuk cluster lokal di Outposts AWS	Anda sekarang dapat membuat klaster lokal Amazon EKS di Outpost menggunakan versi Kubernetes —. 1.22 1.25	1 Maret 2023
Versi Kubernetes 1.25	Menambahkan 1.25 dukungan versi Kubernetes untuk klaster baru dan peningkatan versi.	22 Februari 2023
AWS pembaruan kebijakan terkelola - Pembaruan ke kebijakan yang ada	Amazon EKS memperbarui kebijakan AWS terkelola yang ada.	7 Februari 2023
Ekspansi AWS Wilayah Amazon EKS	Amazon EKS sekarang tersedia di Asia Pasifik (Hyderabad) (ap-south-2), Eropa (Zurich) (eu-centra 1-2), dan Eropa (Spanyol) (). eu-south-2 AWS	6 Februari 2023
Versi Kubernetes 1.21 — sekarang 1.24 tersedia untuk cluster lokal di Outposts. AWS	Anda sekarang dapat membuat klaster lokal Amazon EKS di Outpost menggunakan versi Kubernetes —. 1.21 1.24 Sebelumnya, hanya versi 1.21 yang tersedia.	Januari 17, 2023
Amazon EKS sekarang mendukung AWS PrivateLink	Anda dapat menggunakan file AWS PrivateLink untuk membuat koneksi pribadi antara VPC Anda dan Amazon EKS.	16 Desember 2022

Dukungan Windows grup node terkelola	Anda sekarang dapat menggunakan Windows untuk grup node terkelola Amazon EKS.	15 Desember 2022
Add-on Amazon EKS dari vendor perangkat lunak independen sekarang tersedia di Marketplace AWS	Anda sekarang dapat menelusuri dan berlangganan add-on Amazon EKS dari vendor perangkat lunak independen melalui Marketplace. AWS	28 November 2022
AWS pembaruan kebijakan terkelola - Pembaruan ke kebijakan yang ada	Amazon EKS memperbarui kebijakan AWS terkelola yang ada.	17 November 2022
Versi Kubernetes 1.24	Menambahkan 1.24 dukungan versi Kubernetes untuk kluster baru dan peningkatan versi.	15 November 2022
Ekspansi AWS Wilayah Amazon EKS	Amazon EKS sekarang tersedia di AWS Wilayah Timur Tengah (UEAme-central-1).	November 3, 2022
AWS pembaruan kebijakan terkelola - Pembaruan ke kebijakan yang ada	Amazon EKS memperbarui kebijakan AWS terkelola yang ada.	24 Oktober 2022
AWS pembaruan kebijakan terkelola - Pembaruan ke kebijakan yang ada	Amazon EKS memperbarui kebijakan AWS terkelola yang ada.	20 Oktober 2022
Cluster lokal di AWS Outposts sekarang tersedia	Anda sekarang dapat membuat cluster lokal Amazon EKS di Outpost.	19 September 2022

Kuota berbasis vCPU Fargate	Fargate sedang bertransisi dari kuota berbasis Pod ke kuota berbasis vCPU.	8 September 2022
AWS pembaruan kebijakan terkelola - Pembaruan ke kebijakan yang ada	Amazon EKS memperbarui kebijakan AWS terkelola yang ada.	31 Agustus 2022
Pemantauan biaya	Amazon EKS sekarang mendukung Kubecost, yang memungkinkan Anda memantau biaya yang dipecah berdasarkan sumber daya Kubernetes termasuk Pod, node, namespace, dan label.	Agustus 24, 2022
AWS pembaruan kebijakan terkelola - Kebijakan baru	Amazon EKS menambahkan kebijakan AWS terkelola baru.	Agustus 24, 2022
AWS pembaruan kebijakan terkelola - Kebijakan baru	Amazon EKS menambahkan kebijakan AWS terkelola baru.	23 Agustus 2022
Tag sumber daya untuk penagihan	Menambahkan dukungan tag alokasi biaya yang <code>aws:eks:cluster-name</code> dihasilkan untuk semua cluster.	Agustus 16, 2022
Wildcard profil Fargate	Menambahkan dukungan untuk wildcard profil Fargate dalam kriteria pemilih untuk ruang nama, kunci label, dan nilai label.	Agustus 16, 2022
Versi Kubernetes 1.23	Menambahkan 1.23 dukungan versi Kubernetes untuk klaster baru dan peningkatan versi.	Agustus 11, 2022

Lihat sumber daya Kubernetes di AWS Management Console	Anda sekarang dapat melihat informasi tentang sumber daya Kubernetes yang diterapkan ke kluster Anda menggunakan AWS Management Console	Mei 3, 2022
Ekspansi AWS Wilayah Amazon EKS	Amazon EKS sekarang tersedia di AWS Wilayah Asia Pasifik (Jakarta) (ap-southeast-3).	2 Mei 2022
Halaman observabilitas dan dukungan add-on ADOT	Ditambahkan halaman Observability dan AWS Distro untuk OpenTelemetry (ADOT).	21 April 2022
Versi Kubernetes 1.22	Menambahkan 1.22 dukungan versi Kubernetes untuk kluster baru dan peningkatan versi.	4 April 2022
AWS pembaruan kebijakan terkelola - Kebijakan baru	Amazon EKS menambahkan kebijakan AWS terkelola baru.	4 April 2022
Menambahkan rincian patch Fargate Pod	Saat memutakhirkan Pod Fargate, Amazon EKS pertama-tama mencoba mengusir Pod berdasarkan anggaran gangguan Pod Anda. Anda dapat membuat aturan acara untuk bereaksi terhadap pengusuran yang gagal sebelum Pod dihapus.	1 April 2022

Rilis penuh: Dukungan add-on untuk driver Amazon EBS CSI	Anda sekarang dapat menggunakan AWS Management Console, AWS CLI, dan API untuk mengelola driver Amazon EBS CSI.	31 Maret 2022
AWS Update konten Outposts	Petunjuk untuk menyebarkan kluster Amazon EKS di AWS Outposts.	22 Maret 2022
AWS pembaruan kebijakan terkelola - Pembaruan ke kebijakan yang ada	Amazon EKS memperbarui kebijakan AWS terkelola yang ada.	Maret 21, 2022
containerd Dukungan Windows	Anda sekarang dapat memilih containerd runtime untuk node Windows.	Maret 14, 2022
Menambahkan pertimbangan Amazon EKS Connector ke dokumentasi keamanan	Menjelaskan model tanggung jawab bersama yang berkaitan dengan cluster yang terhubung.	25 Februari 2022
Tetapkan IPv6 alamat ke Pod dan layanan	Sekarang Anda dapat membuat kluster 1.21 atau yang lebih baru yang menetapkan IPv6 alamat ke Pod dan layanan Anda.	6 Januari 2022
AWS pembaruan kebijakan terkelola - Pembaruan ke kebijakan yang ada	Amazon EKS memperbarui kebijakan AWS terkelola yang ada.	13 Desember 2021
Rilis pratinjau: Dukungan add-on untuk driver Amazon EBS CSI	Anda sekarang dapat melihat pratinjau menggunakan AWS Management Console, AWS CLI, dan API untuk mengelola driver Amazon EBS CSI.	Desember 9, 2021

<u>Dukungan autoscaler Karpenter</u>	Anda sekarang dapat menggunakan proyek open-source Karpenter untuk menskalakan otomatis node Anda.	29 November 2021
<u>Dukungan filter Kubernetes Bit Lancar di logging Fargate</u>	Anda sekarang dapat menggunakan filter Fluent Bit Kubernetes dengan logging Fargate.	November 10, 2021
<u>Dukungan Windows tersedia di bidang kontrol</u>	Dukungan Windows sekarang tersedia di bidang kontrol Anda. Anda tidak perlu lagi mengaktifkannya di pesawat data Anda.	November 9, 2021
<u>Bottlerocket ditambahkan sebagai tipe AMI untuk grup node terkelola</u>	Sebelumnya, Bottlerocket hanya tersedia sebagai opsi node yang dikelola sendiri. Sekarang dapat dikonfigurasi sebagai grup node terkelola , mengurangi upaya yang diperlukan untuk memenuhi persyaratan kepatuhan node.	28 Oktober 2021
<u>DL1 dukungan pengemudi</u>	Custom Amazon Linux AMIs sekarang mendukung beban kerja pembelajaran mendalam untuk Amazon Linux 2. Pengaktifan ini memungkinkan konfigurasi dasar lokal atau cloud generik.	25 Oktober 2021

VT1 dukungan video	Custom Amazon Linux AMIs sekarang mendukung VT1 untuk beberapa distribusi. Pengaktifan ini mengiklankan perangkat Xilinx U30 di cluster Amazon EKS Anda.	13 September 2021
Konektor Amazon EKS sekarang tersedia	Anda dapat menggunakan Amazon EKS Connector untuk mendaftarkan dan menghubungkan klaster Kubernetes yang sesuai dan AWS memvisualisasikannya di konsol Amazon EKS.	8 September 2021
Amazon EKS Anywhere sekarang tersedia	Amazon EKS Anywhere adalah opsi penerapan baru untuk Amazon EKS yang dapat Anda gunakan untuk membuat dan mengoperasikan klaster Kubernetes di lokasi.	8 September 2021
Amazon FSx untuk driver NetApp ONTAP CSI	Menambahkan topik yang merangkum Amazon FSx untuk driver NetApp ONTAP CSI dan memberikan tautan ke referensi lain.	2 September 2021
Grup node terkelola sekarang secara otomatis menghitung Pod maksimum yang direkomendasikan Amazon EKS untuk node	Grup node terkelola sekarang menghitung otomatis Pod maksimum Amazon EKS untuk node yang Anda gunakan tanpa template peluncuran, atau dengan template peluncuran yang belum Anda tentukan ID AMI.	Agustus 30, 2021

[Hapus pengelolaan pengaturan add-on Amazon EKS tanpa menghapus perangkat lunak add-on Amazon EKS](#)

Anda sekarang dapat menghapus add-on Amazon EKS tanpa menghapus perangkat lunak add-on dari cluster Anda.

Agustus 20, 2021

[Buat Pod multi-homed menggunakan Multus](#)

Anda sekarang dapat menambahkan beberapa antarmuka jaringan ke Pod menggunakan Multus.

2 Agustus 2021

[Tambahkan lebih banyak alamat IP ke EC2 node Amazon Linux Anda](#)

Anda sekarang dapat menambahkan lebih banyak alamat IP secara signifikan ke EC2 node Amazon Linux Anda. Ini berarti Anda dapat menjalankan kepadatan Pod yang lebih tinggi pada setiap node. Anda sekarang dapat menambahkan lebih banyak alamat IP secara signifikan ke EC2 node Amazon Linux Anda. Ini berarti Anda dapat menjalankan kepadatan Pod yang lebih tinggi pada setiap node.

27 Juli 2021

<u>containerd runtime bootstrap</u>	Amazon EKS yang dioptimalkan Amazon Linux Amazon Amazon Machine Image (AMI) sekarang berisi flag bootstrap yang dapat Anda gunakan untuk mengaktifkan containerd runtime di Amazon EKS dioptimalkan dan AMIs Bottlerocket. Bendera ini tersedia di semua AMI versi Kubernetes yang didukung.	19 Juli 2021
<u>Versi Kubernetes 1.21</u>	Menambahkan dukungan versi Kubernetes. 1.21	19 Juli 2021
<u>Ditambahkan topik kebijakan terkelola</u>	Daftar semua kebijakan dan perubahan yang dikelola Amazon EKS IAM yang dibuat sejak 17 Juni 2021.	17 Juni 2021
<u>Gunakan grup keamanan untuk Pod dengan Fargate</u>	Anda sekarang dapat menggunakan grup keamanan untuk Pod dengan Fargate, selain menggunakannya dengan node Amazon EC2 .	1 Juni 2021
<u>Menambahkan add-on CoreDNS kube-proxy dan Amazon EKS</u>	Amazon EKS sekarang dapat membantu Anda mengelola CoreDNS dan add-on Amazon EKS kube-proxy untuk kluster Anda.	19 Mei 2021
<u>Versi Kubernetes 1.20</u>	Menambahkan 1.20 dukungan versi Kubernetes untuk kluster baru dan peningkatan versi.	18 Mei 2021

AWS Load Balancer Controller dirilis 2.2.0	Anda sekarang dapat menggunakan AWS Load Balancer Controller untuk membuat Elastic Load Balancer menggunakan target instance atau IP.	14 Mei 2021
Node taints untuk grup node terkelola	Amazon EKS sekarang mendukung penambahan taint simpul ke grup-grup simpul terkelola.	11 Mei 2021
Enkripsi rahasia untuk cluster yang ada	Amazon EKS sekarang mendukung penambahan enkripsi rahasia ke cluster yang ada.	26 Februari 2021
Versi Kubernetes 1.19	Menambahkan 1.19 dukungan versi Kubernetes untuk kluster baru dan peningkatan versi.	16 Februari 2021
Amazon EKS sekarang mendukung penyedia identitas OpenID Connect (OIDC) sebagai metode untuk mengautentikasi pengguna ke versi atau cluster yang lebih baru. 1.16	Penyedia identitas OIDC dapat digunakan dengan, atau sebagai alternatif untuk AWS Identity and Access Management (IAM).	12 Februari 2021
Lihat sumber daya node dan beban kerja di AWS Management Console	Anda sekarang dapat melihat detail tentang node terkelola, dikelola sendiri, dan Fargate serta beban kerja Kubernetes yang Anda gunakan di AWS Management Console	1 Desember 2020

<u>Menerapkan jenis Instance Spot dalam grup node terkelola</u>	Anda sekarang dapat men-deploy beberapa jenis Spot atau Instans Sesuai Permintaan ke grup simpul terkelola.	1 Desember 2020
<u>Amazon EKS sekarang dapat mengelola add-on khusus untuk klaster Anda</u>	Anda dapat mengelola sendiri add-on, atau mengizinkan Amazon EKS mengontrol peluncuran dan versi add-on melalui Amazon EKS API.	1 Desember 2020
<u>Bagikan ALB di beberapa Ingress</u>	Anda sekarang dapat berbagi AWS Application Load Balancer (ALB) di beberapa Kubernetes Ingresses. Di masa lalu, Anda harus menerapkan ALB terpisah untuk setiap Ingress.	23 Oktober 2020
<u>Dukungan target IP NLB</u>	Anda sekarang dapat menerapkan Network Load Balancer dengan target IP. Ini berarti Anda dapat menggunakan NLB untuk memuat lalu lintas jaringan keseimbangan ke Pod Fargate dan langsung ke Pod yang berjalan di node Amazon. EC2	23 Oktober 2020
<u>Versi Kubernetes 1.18</u>	Menambahkan 1.18 dukungan versi Kubernetes untuk klaster baru dan peningkatan versi.	13 Oktober 2020

Tentukan blok CIDR kustom untuk penetapan alamat IP layanan Kubernetes.	Anda sekarang dapat menentukan blok CIDR kustom tempat Kubernetes memberikan alamat IP layanan.	29 September 2020
Menetapkan grup keamanan ke Pod individual	Sekarang Anda dapat mengaitkan grup keamanan yang berbeda ke beberapa Pod individual yang berjalan di banyak jenis EC2 instans Amazon.	9 September 2020
Menyebarkan Bottlerocket di node Anda	Anda sekarang dapat menyebarkan node yang menjalankan Bottlerocket .	31 Agustus 2020
Kemampuan untuk meluncurkan node Arm umumnya tersedia	Sekarang Anda dapat meluncurkan simpul Arm di grup simpul terkelola dan dikelola sendiri.	17 Agustus 2020
Templat peluncuran grup node terkelola dan AMI khusus	Sekarang Anda dapat menerapkan grup node terkelola yang menggunakan template EC2 peluncuran Amazon. Peluncuran templat dapat menentukan kustom AMI, jika Anda mau.	17 Agustus 2020
Dukungan EFS untuk AWS Fargate	Anda sekarang dapat menggunakan Amazon EFS dengan AWS Fargate.	17 Agustus 2020

Pembaruan versi platform Amazon EKS	Ini adalah versi platform baru dengan perbaikan dan peningkatan keamanan. Ini termasuk dukungan UDP untuk jenis layanan LoadBalancer saat menggunakan Network Load Balancer dengan versi Kubernetes atau yang lebih baru. 1.15 Untuk informasi selengkapnya, lihat masalah Izinkan UDP untuk AWS Network Load Balancer. GitHub	12 Agustus 2020
Ekspansi AWS Wilayah Amazon EKS	Amazon EKS sekarang tersedia di AWS Wilayah Afrika (Cape Town) (af-south-1) dan Eropa (Milaneu-south-1) ().	6 Agustus 2020
Metrik penggunaan Fargate	AWS Fargate menyediakan metrik CloudWatch penggunaan yang memberikan visibilitas ke dalam penggunaan sumber daya Fargate On-Demand akun Anda.	3 Agustus 2020
Versi Kubernetes 1.17	Menambahkan 1.17 dukungan versi Kubernetes untuk kluster baru dan peningkatan versi.	10 Juli 2020

Membuat dan mengelola resource App Mesh dari dalam Kubernetes dengan pengontrol App Mesh untuk Kubernetes	Anda dapat membuat dan mengelola sumber daya App Mesh dari dalam Kubernetes. Kontroler juga secara otomatis menyuntikkan proxy Envoy dan kontainer init ke dalam Pod yang Anda deploy.	18 Juni 2020
Amazon EKS sekarang mendukung node Amazon EC2 Inf1	Anda dapat menambahkan node Amazon EC2 Inf1 ke cluster Anda.	4 Juni 2020
Ekspansi AWS Wilayah Amazon EKS	Amazon EKS sekarang tersedia di Wilayah AWS GovCloud (AS-Timur) (us-gov-east-1) dan AWS GovCloud (AS-Barat) (us-gov-west-1) AWS .	13 Mei 2020
Kubernetes tidak lagi 1.12 didukung di Amazon EKS	Versi Kubernetes tidak lagi 1.12 didukung di Amazon EKS. Perbarui 1.12 cluster apa pun ke versi 1.13 atau yang lebih baru untuk menghindari gangguan layanan.	12 Mei 2020
Versi Kubernetes 1.16	Menambahkan 1.16 dukungan versi Kubernetes untuk klaster baru dan peningkatan versi.	30 April 2020
Menambahkan peran AWSServiceRoleForAmazonEKS terkait layanan EKS	Menambahkan peran terkait layanan AWSServiceRoleForAmazonEKS.	16 April 2020

Versi Kubernetes 1.15	Menambahkan 1.15 dukungan versi Kubernetes untuk kluster baru dan peningkatan versi.	10 Maret 2020
Ekspansi AWS Wilayah Amazon EKS	Amazon EKS sekarang tersedia di AWS Wilayah Beijing (cn-north-1) dan Ningxia (cn-northwest-1).	26 Februari 2020
FSx untuk driver Lustre CSI	Menambahkan topik untuk menginstal driver FSx for Lustre CSI di cluster Kubernetes Amazon EKS. 1.14	23 Desember 2019
Membatasi akses jaringan ke titik akhir akses publik kluster	Dengan pembaruan ini, Anda dapat menggunakan Amazon EKS untuk membatasi rentang CIDR yang dapat berkomunikasi ke titik akhir akses publik server Kubernetes API.	20 Desember 2019
Menyelesaikan alamat titik akhir akses pribadi untuk kluster dari luar VPC	Dengan pembaruan ini, Anda dapat menggunakan Amazon EKS untuk menyelesaikan titik akhir akses pribadi server API Kubernetes dari luar VPC.	13 Desember 2019
(Beta) Node EC2 instans Amazon EC2 A1 Amazon	Luncurkan node EC2 instans Amazon EC2 A1 Amazon yang mendaftar dengan cluster Amazon EKS Anda.	4 Desember 2019
Membuat cluster di AWS Outposts	Amazon EKS sekarang mendukung pembuatan cluster di AWS Outposts.	3 Desember 2019

AWS Fargate di Amazon EKS	Cluster Amazon EKS Kubernetes sekarang mendukung menjalankan Pod di Fargate.	3 Desember 2019
Ekspansi AWS Wilayah Amazon EKS	Amazon EKS sekarang tersedia di Wilayah Kanada (Tengah) <code>ca-central-1</code> (AWS).	21 November 2019
Grup simpul terkelola	Grup node terkelola Amazon EKS mengotomatiskan penyediaan dan pengelolaan siklus hidup node (instans Amazon EC2) untuk klaster Amazon EKS Kubernetes.	18 November 2019
Pembaruan versi platform Amazon EKS	Versi platform baru yang perlu ditangani CVE-2019-11253 .	6 November 2019
Kubernetes tidak lagi 1.11 didukung di Amazon EKS	Versi Kubernetes tidak lagi 1.11 didukung di Amazon EKS. Harap perbarui 1.11 cluster apa pun ke versi 1.12 atau lebih tinggi untuk menghindari gangguan layanan.	4 November 2019
Ekspansi AWS Wilayah Amazon EKS	Amazon EKS sekarang tersedia di Wilayah Amerika Selatan (São Paulo) (<code>sa-east-1</code>) AWS .	16 Oktober 2019
Dukungan Windows	Cluster Amazon EKS yang menjalankan versi Kubernetes 1.14 sekarang mendukung beban kerja Windows.	7 Oktober 2019

Penskalaan otomatis	Penambahan Sebuah bab untuk menjelaskan beberapa jenis penskalaan otomatis Kubernetes yang didukung pada kluster Amazon EKS.	30 September 2019
Pembaruan Dasbor Kubernetes	Topik yang diperbarui untuk menginstal Dasbor Kubernetes di kluster Amazon EKS untuk menggunakan versi beta. 2.0	28 September 2019
Pengemudi Amazon EFS CSI	Menambahkan topik untuk menginstal driver Amazon EFS CSI di kluster 1.14 Amazon EKS Kubernetes.	19 September 2019
Parameter Amazon EC2 Systems Manager untuk ID AMI Amazon EKS yang dioptimalkan	Menambahkan topik untuk mengambil ID AMI Amazon EKS yang dioptimalkan menggunakan parameter Amazon EC2 Systems Manager. Parameter menghilangkan kebutuhan bagi Anda untuk mencari AMI IDs.	18 September 2019
Penandaan sumber daya Amazon EKS	Anda dapat mengelola penandaan kluster Amazon EKS Anda.	16 September 2019
Pengemudi Amazon EBS CSI	Menambahkan topik untuk menginstal driver Amazon EBS CSI di kluster Amazon EKS Kubernetes. 1.14	9 September 2019

Amazon EKS baru yang dioptimalkan AMI ditambal untuk CVE-2019-9512 dan CVE-2019-9514	Amazon EKS telah memperbarui AMI yang dioptimalkan dengan Amazon EKS untuk mengatasi CVE-2019-9512 dan CVE-2019-9514 .	6 September 2019
Mengumumkan penghentian Kubernetes di Amazon EKS 1.11	Amazon EKS menghentikan dukungan untuk versi Kubernetes 1.11 pada 4 November 2019.	4 September 2019
Versi Kubernetes 1.14	Menambahkan 1.14 dukungan versi Kubernetes untuk kluster baru dan peningkatan versi.	3 September 2019
Peran IAM untuk akun layanan	Dengan IAM role untuk akun layanan di kluster Amazon EKS, Anda dapat mengaitkan IAM role dengan akun layanan Kubernetes. Dengan fitur ini, Anda tidak perlu lagi memberikan izin tambahan ke peran IAM node. Dengan cara ini, Pod pada node itu dapat memanggil AWS APIs.	3 September 2019
Ekspansi AWS Wilayah Amazon EKS	Amazon EKS sekarang tersedia di Wilayah Timur Tengah (Bahrain) (me-south-1) AWS.	29 Agustus 2019
Pembaruan versi platform Amazon EKS	Versi platform baru yang perlu ditangani CVE-2019-9512 dan CVE-2019-9514 .	28 Agustus 2019

Pembaruan versi platform Amazon EKS	Versi platform baru yang perlu ditanggapi CVE-2019-11247 dan CVE-2019-11249 .	5 Agustus 2019
Ekspansi Wilayah Amazon EKS	Amazon EKS sekarang tersedia di AWS Wilayah Asia Pasifik (Hong Kong) (ap-east-1).	31 Juli 2019
Kubernetes 1.10 tidak lagi didukung di Amazon EKS	Versi Kubernetes tidak lagi 1.10 didukung di Amazon EKS. Perbarui 1.10 cluster apa pun ke versi 1.11 atau lebih tinggi untuk menghindari gangguan layanan.	30 Juli 2019
Ditambahkan topik pada ALB ingress controller	AWS ALB Ingress Controller untuk Kubernetes adalah controller yang menyebabkan ALB dibuat ketika resource ingress dibuat.	11 Juli 2019
Amazon EKS baru yang dioptimalkan AMI	Menghapus kubect1 biner yang tidak perlu dari AMIs.	3 Juli 2019
Versi Kubernetes 1.13	Menambahkan 1.13 dukungan versi Kubernetes untuk klaster baru dan peningkatan versi.	Selasa, 18 Juni 2019
Amazon EKS baru yang dioptimalkan AMI ditambah AWS-2019-005	Amazon EKS telah memperbarui AMI Amazon EKS yang dioptimalkan untuk mengatasi kerentanan yang dijelaskan dalam tautan: security/security-buletins/-2019-005/ [AWS-2019-005, type="marketing "].AWS	17 Juni 2019

[Mengumumkan penghentian dukungan 1.10 Kubernetes di Amazon EKS](#)

Amazon EKS berhenti mendukung versi Kubernetes 1.10 pada 22 Juli 2019.

21 Mei 2019

[Pembaruan versi platform Amazon EKS](#)

Versi platform baru untuk Kubernetes 1.11 dan 1.10 cluster untuk mendukung nama DNS kustom dalam sertifikat dan meningkatkan kinerja. kubelet etcd

21 Mei 2019

[Memulai dengan eksctl](#)

Panduan memulai ini menjelaskan bagaimana Anda dapat menginstal semua sumber daya yang diperlukan untuk memulai menggunakan Amazon EKSeksctl. Ini adalah utilitas baris perintah sederhana untuk membuat dan mengelola cluster Kubernetes di Amazon EKS.

10 Mei 2019

[AWS Perintah CLI get-token](#)

`aws eks get-token`
Perintah ditambahkan ke AWS CLI. Anda tidak perlu lagi menginstal AWS IAM Authenticator untuk Kubernetes untuk membuat token keamanan klien untuk komunikasi server API cluster. Tingkatkan instalasi AWS CLI Anda ke versi terbaru untuk menggunakan fungsi baru ini. Untuk informasi selengkapnya, lihat [Menginstal Antarmuka Baris AWS](#) [AWS Perintah](#) di Panduan Pengguna Antarmuka Baris Perintah.

10 Mei 2019

[Pembaruan versi platform Amazon EKS](#)

Versi platform baru untuk 1.12 kluster Kubernetes untuk mendukung nama DNS kustom dalam sertifikat dan meningkatkan kinerja. kubelet etcd Ini memperbaiki bug yang menyebabkan kubelet daemon node meminta sertifikat baru setiap beberapa detik.

8 Mei 2019

[Prometheus tutorial](#)

Penambahan topik untuk mendeploy Prometheus ke kluster Amazon EKS Anda.

5 April 2019

Amazon EKS mengontrol pencatatan pesawat	Dengan pembaruan ini, Anda bisa mendapatkan log audit dan diagnostik langsung dari panel kontrol Amazon EKS. Anda dapat menggunakan CloudWatch log ini di akun Anda sebagai referensi untuk mengamankan dan menjalankan cluster.	4 April 2019
Versi Kubernetes 1.12	Menambahkan 1.12 dukungan versi Kubernetes untuk klaster baru dan peningkatan versi.	28 Maret 2019
Menambahkan panduan memulai App Mesh	Penambahan dokumentasi untuk memulai dengan App Mesh dan Kubernetes.	27 Maret 2019
Akses pribadi titik akhir server API Amazon EKS	Menambahkan dokumentasi untuk menonaktifkan akses publik untuk titik akhir server Kubernetes API klaster Amazon EKS Anda.	19 Maret 2019
Menambahkan topik untuk menginstal Kubernetes Metrics Server	Kubernetes Metrics Server adalah agregator data penggunaan sumber daya di klaster Anda.	18 Maret 2019
Ditambahkan daftar proyek open source terkait	Proyek-proyek open source ini memperluas fungsionalitas klaster Kubernetes yang berjalan AWS, termasuk cluster yang dikelola oleh Amazon EKS.	15 Maret 2019

[Ditambahkan topik untuk menginstal Helm lokal](#)

Pengnedali paket helm untuk Kubernetes membantu Anda menginstal dan mengelola aplikasi di klaster Kubernetes Anda. Topik ini menunjukkan cara menginstal dan menjalankan helm dan tiller binari secara lokal. Dengan begitu, Anda dapat menginstal dan mengelola bagan menggunakan Helm CLI di sistem lokal Anda.

11 Maret 2019

[Pembaruan versi platform Amazon EKS](#)

[Versi platform baru yang memperbarui 1.11 cluster Amazon EKS Kubernetes ke level 1.11.8 patch untuk mengatasi CVE-2019-1002100.](#)

8 Maret 2019

[Peningkatan batas cluster](#)

Amazon EKS telah meningkatkan jumlah cluster yang dapat Anda buat di AWS Wilayah dari 3 menjadi 50.

13 Februari 2019

[Ekspansi AWS Wilayah Amazon EKS](#)

Amazon EKS sekarang tersedia di Eropa (London) (eu-west-2), Eropa (Paris) (eu-west-3), dan Asia Pasifik (Mumbai) (ap-south-1) AWS Wilayah.

13 Februari 2019

[Amazon EKS baru yang dioptimalkan AMI ditambah ALAS-2019-1156](#)

Amazon EKS telah memperbarui AMI Amazon EKS yang dioptimalkan untuk mengatasi kerentanan yang dijelaskan dalam [ALAS-2019-1156](#).

11 Februari 2019

[Amazon EKS baru yang dioptimalkan AMI ditambah ALAS2-2019-1141](#)

Amazon EKS telah memperbarui AMI Amazon EKS yang dioptimalkan untuk mengatasi CVEs yang direferensikan di [ALAS2-2019-1141](#).

9 Januari 2019

[Ekspansi AWS Wilayah Amazon EKS](#)

Amazon EKS sekarang tersedia di AWS Wilayah Asia Pasifik (Seoul) (ap-northeast-2).

9 Januari 2019

[Ekspansi wilayah Amazon EKS](#)

Amazon EKS sekarang tersedia di AWS Wilayah tambahan berikut: Eropa (Frankfurt) (eu-central-1), Asia Pasifik (Tokyo) (ap-northeast-1), Asia Pasifik (Singapura) (ap-southeast-1), dan Asia Pasifik (Sydney) (ap-southeast-2).

19 Desember 2018

[Pembaruan kluster Amazon EKS](#)

Penambahan dokumentasi untuk Amazon EKS [pembaruan versi kluster Kubernetes](#) dan [penggantian simpul](#).

12 Desember 2018

Ekspansi AWS Wilayah Amazon EKS	Amazon EKS sekarang tersedia di AWS Wilayah Eropa (Stockholm) (eu-north-1).	11 Desember 2018
Pembaruan versi platform Amazon EKS	Versi platform baru yang memperbarui Kubernetes ke level patch ke tautan alamat:security/security-bulletins/ 1.10.11 -2018-020 / [AWS CVE-2018-1002105, type="marketing "].	4 Desember 2018
Ditambahkan 1.0.0 dukungan versi untuk ALB ingress controller	Pengontrol ingress ALB merilis versi 1.0.0 dengan dukungan formal dari AWS	20 November 2018
Menambahkan dukungan untuk konfigurasi jaringan CNI	Plugin Amazon VPC CNI untuk versi Kubernetes 1.2.1 sekarang mendukung konfigurasi jaringan kustom untuk antarmuka jaringan Pod sekunder.	16 Oktober 2018
Menambahkan dukungan untuk MutatingAdmissionWebhook dan ValidatingAdmissionWebhook	Amazon EKS versi platform 1.10-eks.2 sekarang mendukung MutatingAdmissionWebhook dan ValidatingAdmissionWebhook pengendali izin masuk.	10 Oktober 2018
Menambahkan informasi AMI mitra	Canonical telah bermitra dengan Amazon EKS untuk membuat node AMIs yang dapat Anda gunakan di cluster Anda.	3 Oktober 2018

Menambahkan instruksi untuk perintah AWS CLI update-kubeconfig	Amazon EKS telah menambahkan update-kubeconfig ke AWS CLI untuk menyederhanakan proses pembuatan kubeconfig file untuk mengakses cluster Anda.	21 September 2018
Amazon EKS baru dioptimalkan AMIs	Amazon EKS telah memperbarui Amazon EKS yang dioptimalkan AMIs (dengan dan tanpa dukungan GPU) untuk menyediakan berbagai perbaikan keamanan dan pengoptimalan AMI.	13 September 2018
Ekspansi AWS Wilayah Amazon EKS	Amazon EKS sekarang tersedia di Wilayah Eropa (Irlandia) eu-west-1 ().	5 September 2018
Pembaruan versi platform Amazon EKS	Versi platform baru dengan dukungan untuk lapisan agregasi Kubernetes dan Horizontal Pod Autoscaler (HPA).	31 Agustus 2018
Amazon EKS baru dioptimalkan AMIs dan dukungan GPU	Amazon EKS telah memperbarui AMI Amazon EKS yang dioptimalkan untuk menggunakan template AWS CloudFormation node baru dan skrip bootstrap . Selain itu, AMI yang dioptimalkan dengan Amazon EKS dengan dukungan GPU yang baru telah tersedia.	22 Agustus 2018

[Amazon EKS baru yang dioptimalkan AMI ditambah ALAS2-2018-1058](#)

Amazon EKS telah memperbarui AMI Amazon EKS yang dioptimalkan untuk mengatasi CVEs yang direferensikan di [ALAS2-2018-1058](#).

14 Agustus 2018

[Amazon EKS mengoptimalkan skrip pembuatan AMI](#)

Amazon EKS telah menjadikan skrip pembuatan sebagai sumber terbuka yang digunakan untuk membangun AMI yang dioptimalkan dengan Amazon EKS. Skrip build ini sekarang tersedia di GitHub.

10 Juli 2018

[Amazon EKS rilis awal](#)

Dokumentasi awal untuk peluncuran layanan

5 Juni 2018

Berkontribusi pada Panduan Pengguna EKS

AWS telah meluncurkan pengalaman kontribusi yang lebih baik untuk Panduan Pengguna EKS.

Anda sekarang dapat mengedit sumber Panduan Pengguna EKS secara langsung di GitHub.

Dokumen sekarang menggunakan AsciiDoc, bahasa penulisan yang kuat yang mirip dengan penurunan harga. AsciiDoc menggabungkan sintaks sederhana dengan fitur dokumentasi perusahaan seperti pemformatan lanjutan, referensi silang, dan kontrol keamanan.

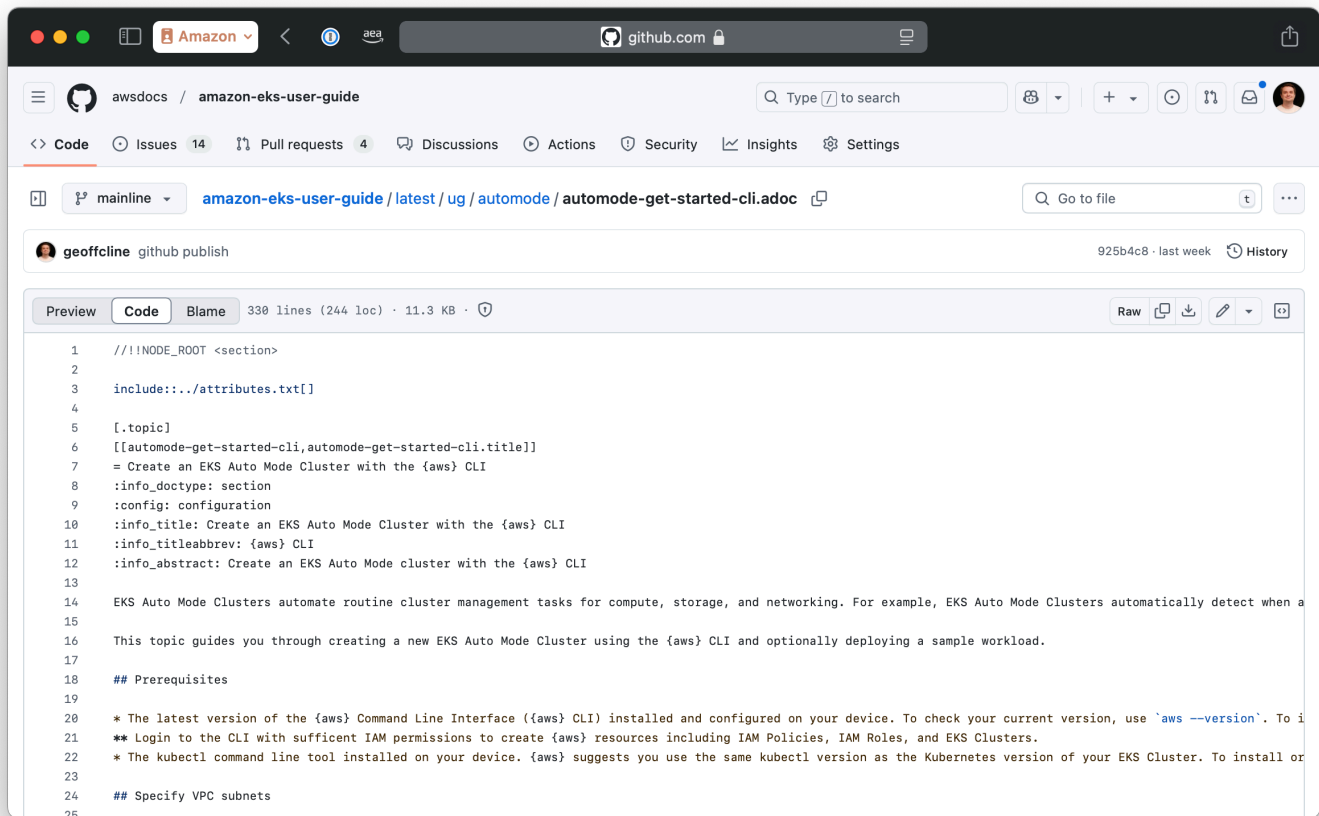
Anda sekarang dapat mengedit Dokumen EKS secara langsung di GitHub. Proses ramping kami meliputi:

- Pemrosesan permintaan tarik lebih cepat
- Mengurangi langkah manual
- Pemeriksaan kualitas konten otomatis

Kami menantikan kontribusi Anda.

Mengedit satu halaman dari browser web

Anda dapat dengan mudah mengedit satu halaman di Panduan Pengguna EKS langsung melalui browser web Anda.



Jika Anda ingin mengedit beberapa halaman dari browser web Anda, lihat [the section called “Edit file dengan GitHub”](#).

Prasyarat

- Halaman dokumen untuk diubah dibuka di browser web.
- Masuk ke GitHub.

Prosedur

1. Arahkan ke halaman yang ingin Anda edit di Panduan Pengguna Amazon EKS.
2. Di panel kanan, pilih GitHub tautan Edit halaman ini.
3. Setelah GitHub aktif, buka editor dengan:
 - Menekan e tombol pada keyboard Anda.
 - Mengklik ikon pensil dan memilih Edit di Tempat dari menu tarik-turun.

- Jika Anda tidak memiliki opsi untuk mengedit, Anda harus masuk ke GitHub. GitHub Akun Anda tidak memerlukan izin khusus untuk menyarankan perubahan. Namun, kontributor Amazon internal harus menautkan profil mereka GitHub .
4. Buat perubahan yang diperlukan pada konten di GitHub editor.
 - Editor menyediakan penyorotan sintaks dan kemampuan pratinjau.
 - Anda dapat menggunakan AsciiDoc markup untuk memformat perubahan Anda.
 - Anda dapat menggunakan `ctrl-f` untuk membuka antarmuka temukan/ganti.
 5. (Opsional) Pratinjau perubahan Anda.
 - Gunakan `preview` tab untuk melihat pratinjau perubahan Anda dengan format kaya.
 - Gunakan `show diff` opsi untuk menyorot bagian yang diubah. Bagian yang dihapus memiliki indikator merah di margin kiri. Bagian baru memiliki indikator hijau di margin kiri.
 6. Setelah selesai mengedit, klik tombol Komit perubahan... di bagian atas editor
 7. Dalam dialog komit:
 - Verifikasi alamat email Anda benar.
 - Tambahkan pesan komit singkat namun deskriptif yang menjelaskan perubahan Anda.
 - Secara opsional tambahkan deskripsi yang lebih panjang jika diperlukan.
 - Pilih untuk membuat cabang baru dan pull request.

Anda telah membuat permintaan tarik termasuk perubahan yang diusulkan.

Ikhtisar permintaan tarik

Saat Anda membuat PR:

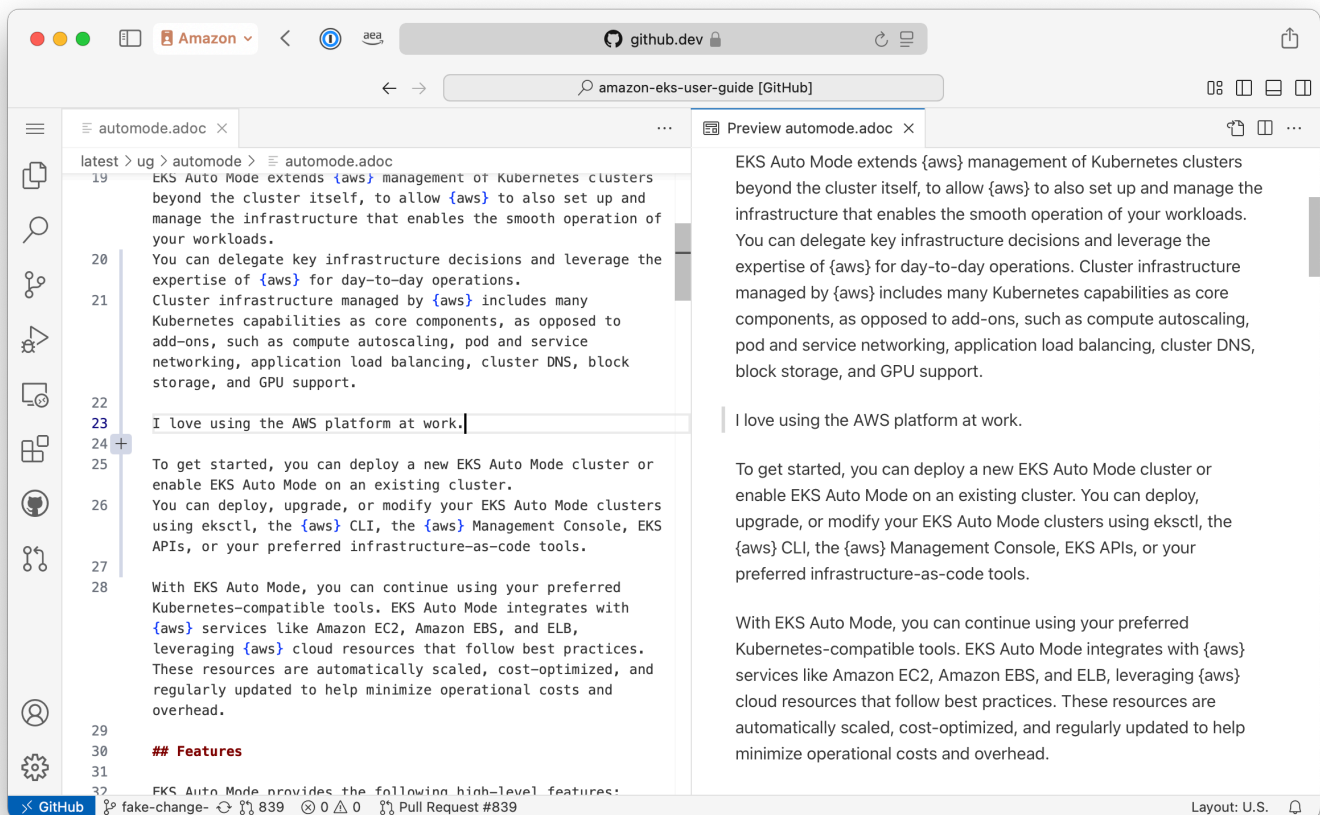
- Perubahan Anda dikirimkan untuk ditinjau oleh pengelola repositori.
- Peninjau dapat mengomentari perubahan Anda dan meminta modifikasi.
- Pengujian otomatis dapat dijalankan untuk memvalidasi perubahan Anda.
- Setelah disetujui, perubahan Anda dapat digabungkan ke dalam repositori utama.

Permintaan tarik membantu memastikan kualitas dan menyediakan cara untuk mendiskusikan perubahan sebelum diintegrasikan.

[Pelajari cara permintaan tarik ditinjau dan disetujui di GitHub Dokumen.](#)

Mengedit beberapa file dari browser web dengan Editor GitHub Web

Jika Anda ingin mengusulkan perubahan ke beberapa halaman, atau membuat halaman dokumen baru, gunakan editor GitHub web.dev. Editor web ini didasarkan pada editor teks Visual Studio Code yang populer.



Prasyarat

- Masuk ke GitHub
- Keakraban dengan editor Kode Visual Studio
- Keakraban dengan Git

Prosedur

Note

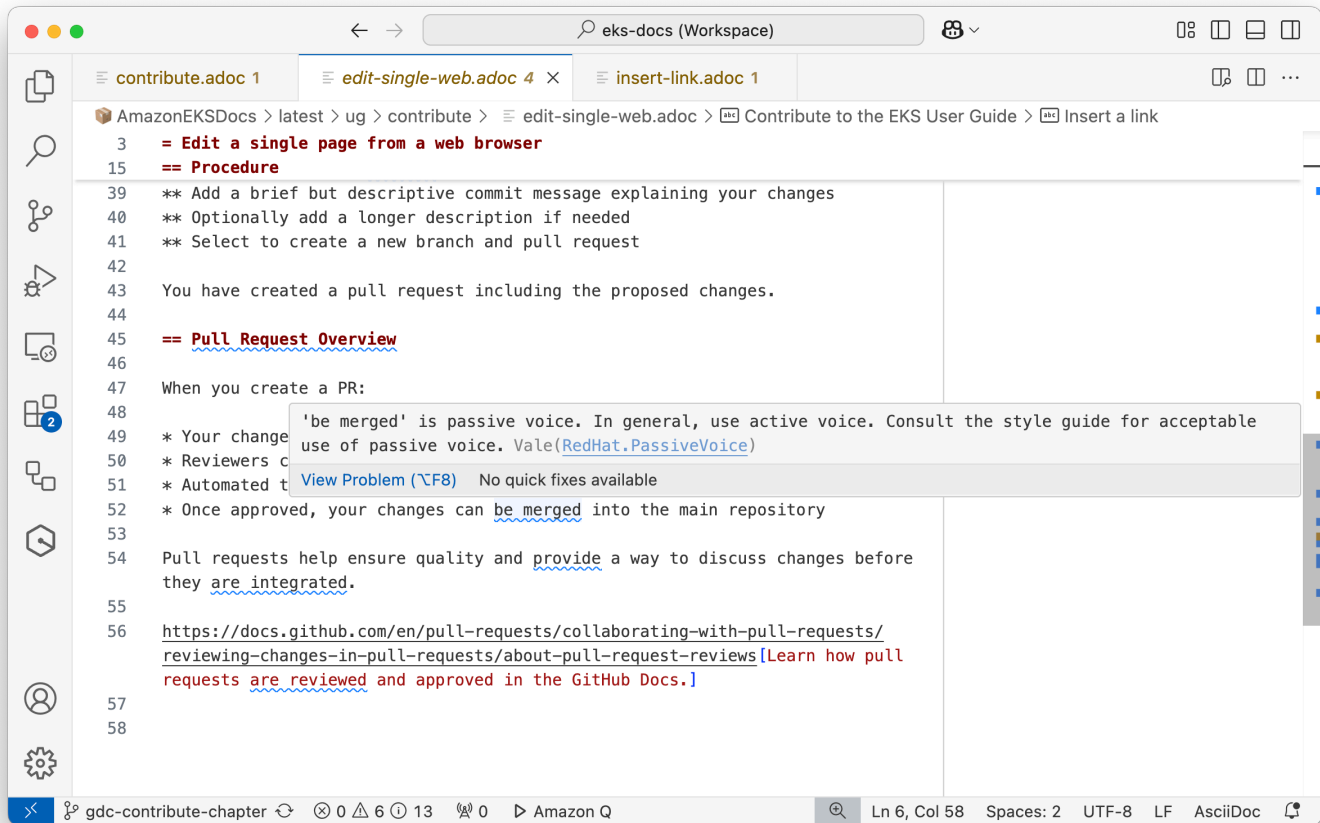
Tim EKS Docs telah membuat file ruang kerja yang menyertakan konfigurasi yang disarankan untuk editor, seperti pembungkus teks dan penyorotan sintaks. AsciiDoc Kami sarankan Anda memuat file ruang kerja ini.

1. Buka [ruang kerja](#) di GitHub .dev.
 - Anda dapat menandai URL <https://github.dev/awsdocs/amazon-eks-user-guide/blob/mainline/eks-docs.code-workspace?workspace=true>
2. (Hanya penyiapan pertama kali) Anda mungkin diminta untuk membuat fork repo di akun Anda sendiri GitHub . Terima prompt ini. Untuk informasi selengkapnya, lihat [Tentang fork](#) di GitHub dokumen.
3. (Hanya pengaturan pertama kali) Terima prompt di kanan bawah untuk menginstal AsciiDoc ekstensi.
4. Arahkan ke konten dokumen di `latest/ug`.
 - File dokumen diatur oleh bagian tingkat atas mereka. Misalnya, halaman di bagian “Keamanan” memiliki file sumber di bawah direktori “security/”.
5. Untuk melihat pratinjau halaman dokumen, gunakan tombol Buka pratinjau ke Samping di kanan atas. Ikon termasuk kaca pembesar kecil.
6. Gunakan tab Source Control di sebelah kiri untuk melakukan perubahan Anda. Untuk informasi selengkapnya, lihat dokumen Visual Studio Code:
 - [Komit perubahan](#)
 - [Buat permintaan tarik](#)

Setelah Anda membuat permintaan tarik, permintaan tersebut akan ditinjau oleh tim dokumen.

Lihat umpan balik gaya saat Anda mengetik dengan menginstal Vale secara lokal

Anda dapat melihat umpan balik gaya saat Anda mengetik. Ini membantu mengidentifikasi tulisan dan kesalahan ketik yang canggung.



Ikhtisar:

- CLI Vale memuat panduan gaya dan menjalankannya terhadap file sumber.
- Repo EKS Docs menyertakan file konfigurasi vale yang memuat panduan gaya dan aturan lokal.
- Ekstensi Vale untuk Kode Visual Studio (VS) menampilkan umpan balik vale di dalam editor.

Instal Vale

Ikuti instruksi di dokumen CLI Vale [untuk Menginstal Vale dengan Package Manager](#).

Instal ekstensi VS Code Vale

1. Buka Kode VS.
2. Klik ikon Extensions di Activity Bar (atau tekan Ctrl+Shift+X).
3. Cari “Vale”.

4. Klik Instal pada ekstensi “Vale VSCode” oleh Chris Chinchilla.
5. Muat ulang VS Code saat diminta.

Sinkronisasi Vale

Vale menggunakan file `.vale.ini` konfigurasi di root proyek Anda untuk menentukan aturan gaya mana yang akan diterapkan.

1. Buka Kode VS.
2. Klik Melihat > Terminal (atau tekan `Ctrl+``).
3. Arahkan ke direktori root proyek Anda jika diperlukan.
4. Jalankan perintah :

```
vale sync
```

5. Tunggu Vale selesai mengunduh dan menyinkronkan aturan gaya

Lihat umpan balik gaya di VS Code

1. Buka Markdown atau AsciiDoc file di VS Code.
2. Ekstensi Vale akan secara otomatis memeriksa teks Anda terhadap aturan gaya.
3. Masalah gaya akan digarisbawahi di editor.
4. Arahkan kursor ke teks yang digarisbawahi untuk melihat saran gaya tertentu.
5. Perbaiki masalah dengan mengikuti saran atau berkonsultasi dengan panduan gaya.

Buat halaman baru

Pelajari cara membuat halaman dokumentasi baru. Topik ini mencakup instruksi untuk membuat metadata halaman awal dan menambahkan halaman ke daftar isi panduan.

Buat halaman

1. Arahkan ke direktori chapter. Misalnya, jika Anda ingin membuat halaman baru di bagian “Keamanan”, arahkan ke `latest/ug/security` direktori.

2. Tentukan ID halaman. Berdasarkan konvensi, ID halaman semuanya huruf kecil dan tersegmentasi dengan `-`. ID dari halaman ini adalah `create-page`.
3. Buat file baru dengan ID halaman dan `.adoc` ekstensi. Misalnya, `create-page.adoc`.
4. Masukkan metadata halaman menggunakan template ini:

```
include:../attributes.txt[]

[.topic]
[#unique-page-id]
= Page title
:info_titleabbrev: Short title

[abstract]
--
Brief summary of the page's content.
--

Introduction paragraph goes here.
```

Tambahkan halaman ke navigasi

1. Arahkan ke halaman induk. Halaman induk dari bagian tingkat atas adalah `book.adoc`.
2. Di bagian bawah halaman induk, sertakan halaman anak.

```
include::${filename}[leveloffset=+1]
```

Sebagai contoh:

```
include::create-page.adoc[leveloffset=+1]
```

Sisipkan tautan

AsciiDoc mendukung berbagai jenis tautan. Menggunakan jenis tautan yang tepat adalah penting agar tautan berfungsi dengan baik di lingkungan yang berbeda.

Tautan ke halaman atau bagian dalam Panduan Pengguna EKS

Gunakan referensi silang (xref) untuk menghubungkan antara halaman atau bagian dalam situs dokumentasi yang sama, seperti Panduan Pengguna EKS. Mereka secara otomatis memperbarui jika bagian target bergerak atau diganti namanya.

Gunakan judul halaman sebagai teks tautan

Untuk sebagian besar kasus saat menautkan ke ID lain dalam panduan pengguna ini, gunakan pendekatan berikut agar teks tautan diperbarui secara otomatis ke judul terbaru sesuai kebutuhan.

```
For more information, see <<page-or-section-id>>.
```

Tentukan teks tautan khusus

Untuk kasus di mana Anda harus memiliki teks tautan khusus, gunakan format berikut.

```
Here's an example of a <<page-or-section-id,link with custom text>>.
```

Tautan ke panduan lain di AWS Dokumen

1. Temukan tautan ke halaman AWS dokumentasi.
2. Hapus <https://docs.aws.amazon.com/> awalan, hanya menjaga jalan. Jalan harus dimulai dengan surat.
3. Buat tautan seperti yang ditunjukkan di bawah ini:

```
link:AmazonS3/latest/userguide/create-bucket-overview.html[Create a bucket,  
type="documentation"]
```

Tautan ke halaman web eksternal

Format ini membuat tautan standar ke halaman yang tidak dihosting oleh Amazon. Misalnya, gunakan ini untuk GitHub tautan.

```
https://example.com[Link text]
```

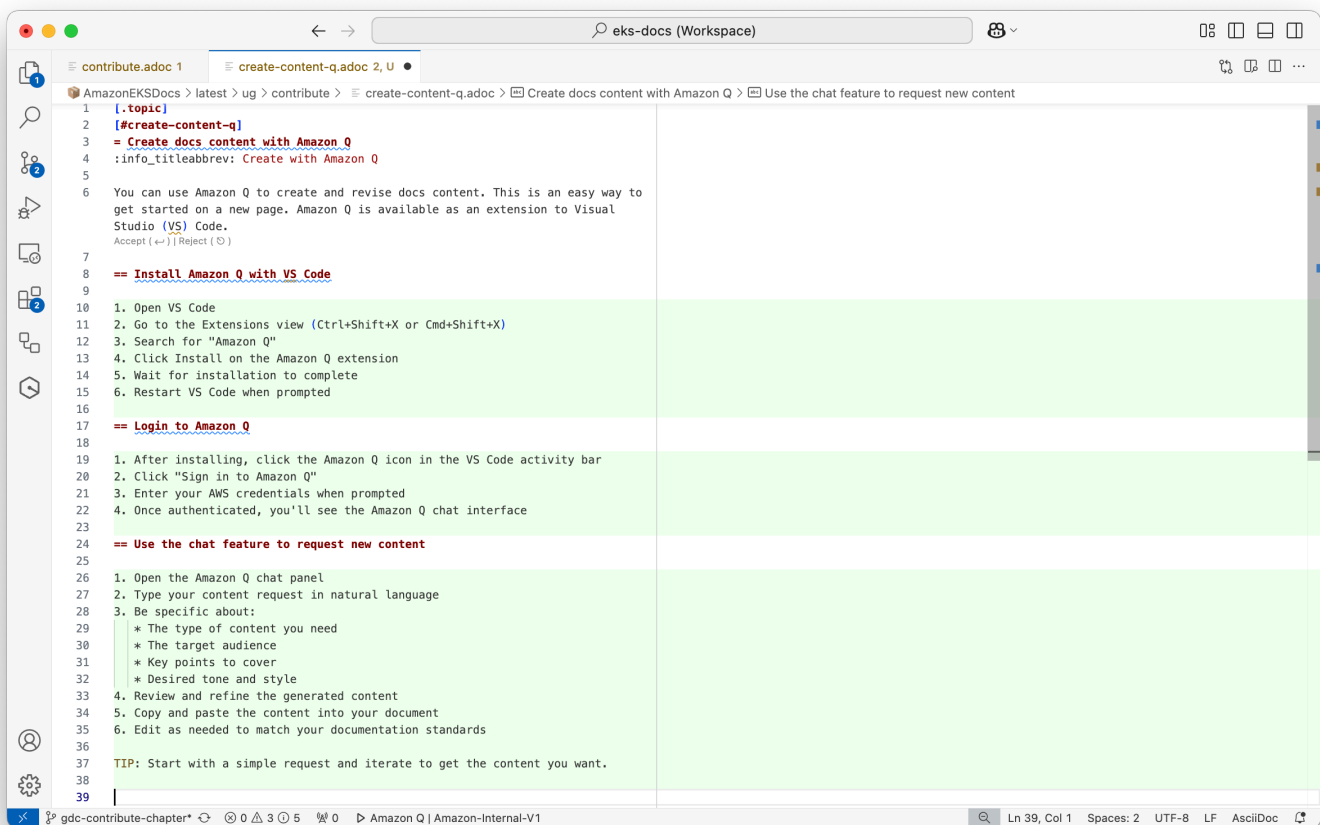
Note

Kami memiliki daftar yang diizinkan untuk domain eksternal. Daftar yang diizinkan ada di `vale/styles/EksDocs/ExternalDomains.yml`

Buat konten dokumen dengan Amazon Q

Anda dapat menggunakan Amazon Q untuk membuat dan merevisi konten dokumen. Ini adalah cara mudah untuk memulai halaman baru. Amazon Q tersedia sebagai ekstensi untuk Visual Studio (VS) Code.

Pada gambar berikut, Amazon Q menghasilkan garis yang ditandai dengan warna hijau.



Instal Amazon Q dengan VS Code

1. Buka Kode VS

2. Pergi ke tampilan Ekstensi (Ctrl+Shift+X atau Cmd+Shift+X)
3. Cari “Amazon Q”
4. Klik Instal pada ekstensi Amazon Q
5. Tunggu instalasi selesai
6. Mulai ulang VS Code saat diminta

Masuk ke Amazon Q

1. Setelah menginstal, pilih ikon Amazon Q di bilah aktivitas VS Code.
2. Pilih Masuk ke Amazon Q.
3. Masukkan AWS kredensi Anda saat diminta.
4. Setelah diautentikasi, Anda akan melihat antarmuka obrolan Amazon Q.

Gunakan Amazon Q untuk membuat konten

1. Buka file yang ingin Anda edit di VS Code.
2. Pilih teks yang ingin Anda revisi atau lokasi untuk konten baru.
3. Tekan Ctrl+I atau Cmd+I.
4. Dalam prompt, spesifik tentang:
 - Jenis konten yang Anda butuhkan.
 - Target audiens.
 - Poin penting untuk dibahas.
 - Nada dan gaya yang diinginkan.
5. Tinjau konten yang dihasilkan dalam pratinjau sebaris.
6. Gunakan enter untuk menerima perubahan, atau esc untuk menolaknya.
7. Edit lebih lanjut sesuai kebutuhan.

Kiat

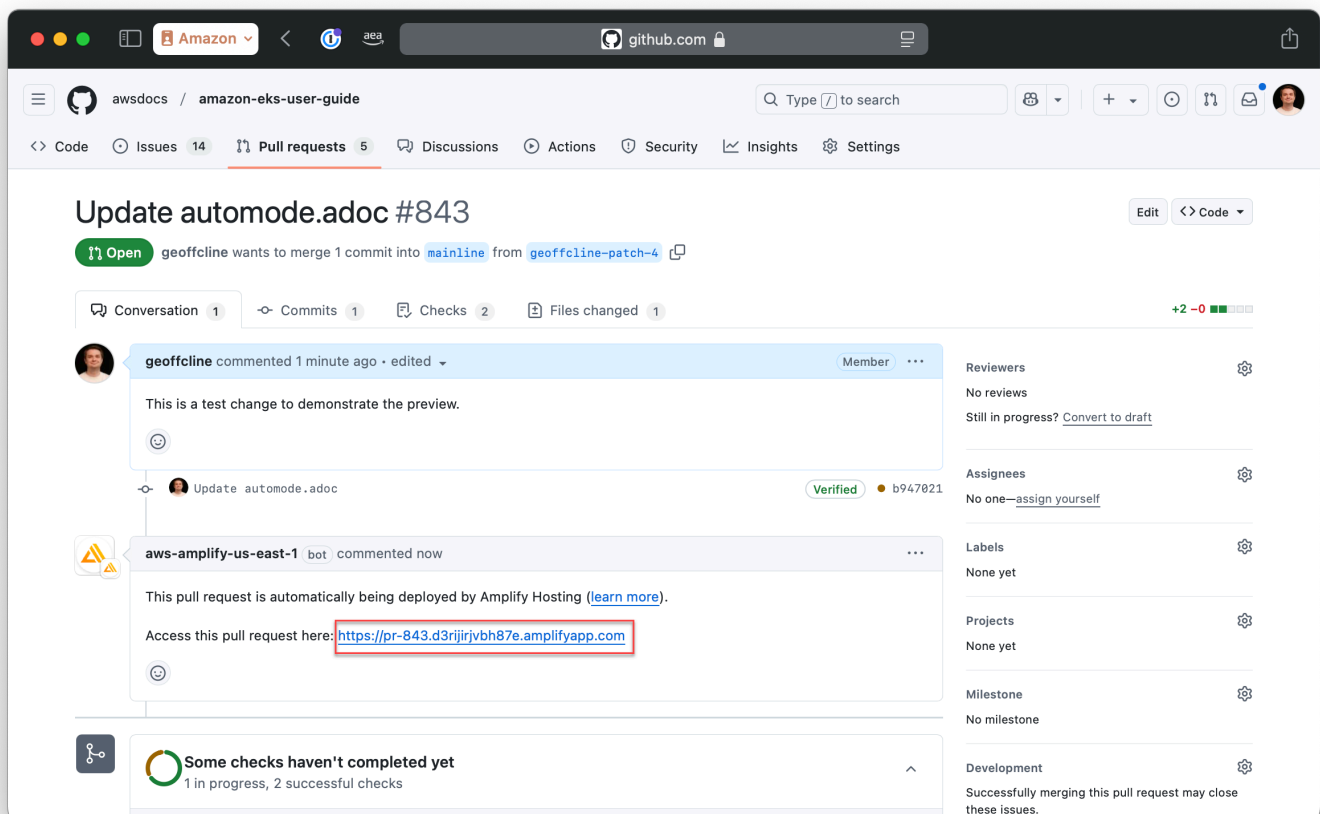
- Mulailah dengan permintaan sederhana dan ulangi untuk mendapatkan konten yang Anda inginkan.
- Buat draf pertama dari judul halaman, lalu minta Q untuk mengisinya.

- Amazon Q mungkin menampilkan Markdown. Ini baik-baik saja. AsciiDoc Perkakas dapat memahami sebagian besar sintaks penurunan harga.

Untuk mempelajari lebih lanjut tentang Pengembang Amazon Q, lihat [Menggunakan Pengembang Amazon Q di IDE](#).

Melihat pratinjau konten permintaan tarik

Panduan Pengguna Amazon EKS GitHub dikonfigurasi untuk membuat dan menghasilkan pratinjau situs dokumen. Pratinjau ini tidak memiliki AWS tema lengkap, tetapi memeriksa konten yang dibangun dengan benar dan tautan berfungsi.



Pratinjau ini di-host di URL sementara oleh AWS Amplify.

Lihat pratinjau

Saat Anda mengirimkan permintaan tarik, AWS Amplify mencoba membuat dan menerapkan pratinjau konten.

Jika build berhasil, `aws-amplify-us-east-1` menambahkan komentar ke permintaan tarik yang memiliki tautan ke pratinjau. Pilih tautan di sebelah kanan “Akses permintaan tarik ini di sini” (seperti yang disebut dalam tangkapan layar dengan garis merah).

Jika build gagal, admin repo dapat melihat log dan memberikan umpan balik.

Note

Jika Anda belum pernah berkontribusi sebelumnya, pengelola proyek mungkin perlu menyetujui menjalankan build.

Batasan pratinjau

Pratinjau dibangun sebagai satu file HTML besar. Ini akan ditampilkan sebagai beberapa halaman saat diterbitkan.

Apa yang berhasil:

- Referensi silang (`xref`)
- Tautan ke internet
- Citra
- Konten yang dihosting dari `samples/`

Apa yang tidak berhasil:

- Tautan ke AWS konten lain, menggunakan `type="documentation"`. Ini karena konten ini tidak ada di lingkungan pratinjau.
- Atribut tidak `{aws}` akan ditampilkan dengan benar. Nilai perubahan ini berdasarkan lingkungan.

AsciiDoc referensi sintaks

AsciiDoc adalah bahasa markup yang disukai untuk AWS dokumentasi. Meskipun tooling memiliki dukungan sebagian untuk sintaks Markdown (seperti judul dan daftar), sebaiknya gunakan AsciiDoc sintaks untuk semua konten untuk memastikan konsistensi dan rendering yang tepat.

Panduan ini mencakup elemen sintaks umum yang Anda perlukan saat berkontribusi pada dokumentasi Amazon EKS. Untuk sintaks yang lebih canggih dan informasi rinci, lihat [AsciiDoc dokumentasi](#).

Halaman baru

Setiap AsciiDoc dokumen baru harus dimulai dengan struktur yang didefinisikan dalam [the section called “Buat halaman”](#).

Judul

Gunakan judul untuk mengatur konten Anda secara hierarkis:

```
= Page/topic title (level 1)
== Section title (level 2)
=== Level 3 heading
==== Level 4 heading
===== Level 5 heading
===== Level 6 heading
```

Catatan: Selalu gunakan kasus kalimat untuk judul dalam AWS dokumentasi.

Pemformatan teks

Format teks untuk menekankan informasi penting:

```
Use *bold text* for UI labels.
Use _italic text_ for introducing terms or light emphasis.
Use `monospace text` for code, file names, and commands.
```

Daftar

Daftar tidak berurutan

Buat daftar berpoin untuk item tanpa urutan tertentu:

```
* First item
* Second item
** Nested item
** Another nested item
* Third item
```

Daftar yang dipesan

Buat daftar bernomor untuk langkah berurutan atau item yang diprioritaskan:

```
. First step
. Second step
.. Substep 1
.. Substep 2
. Third step
```

Tautan

Lihat [the section called “Sisipkan tautan”](#) untuk detail tentang cara memformat tautan dengan benar. Tautan gaya penurunan harga tidak didukung.

Contoh kode

Kode sebaris

Gunakan backticks untuk kode inline:

```
Use the `kubectl get pods` command to list all pods.
```

Blok kode

Buat blok kode dengan penyorotan sintaks dan dukungan untuk atribut (mirip dengan entitas):

```
[source,python,subs="verbatim,attributes"]
----
def hello_world():
    print("Hello, World!")
----
```


Citra

Sisipkan gambar dengan teks alt untuk aksesibilitas:

```
image::images/image-file.png[Alt text description]
```

Tabel

Buat tabel untuk mengatur informasi:

```
[%header,cols="2"]  
|===  
|Header 1  
|Header 2  
  
|Cell 1,1  
|Cell 1,2  
  
|Cell 2,1  
|Cell 2,2  
|===
```

Untuk tabel yang lebih kompleks, lihat [dokumentasi AsciiDoc tabel](#).

Callout

Gunakan callout untuk menyoroti informasi dan peringatan penting:

NOTE: This is a note callout for general information.

TIP: This is a tip callout for helpful advice.

IMPORTANT: This is an important callout for critical information.

Pratinjau:

Note

Ini adalah catatan callout.

Termasuk file lainnya

Sertakan konten dari file lain:

```
include::filename.adoc[]
```

Atribut (mirip dengan entitas)

Gunakan atribut yang telah ditentukan untuk menjaga konsistensi. Secara khusus, Anda HARUS menggunakan atribut untuk AWS dan `arn:aws:`.

```
{aws} provides Amazon EKS as a managed Kubernetes service.
```

```
[source,bash,subs="verbatim,attributes"]
----
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn {arn-aws}iam::aws:policy/AmazonEKSClusterPolicy
----
```

Untuk daftar atribut, lihat di `../attributes.txt` file.

Prosedur

step-by-stepProsedur format:

```
To create an Amaozon EKS cluster. do the following steps.

. Sign in to the {aws} Management Console.
. Open the Amazon EKS console.
. Choose *Create cluster*.

...
```