



Panduan Pengguna

AWS App Mesh



AWS App Mesh: Panduan Pengguna

Copyright © 2026 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 merendahkan atau mendiskreditkan Amazon. Semua merek dagang lain yang tidak dimiliki oleh Amazon merupakan hak milik masing-masing pemiliknya, yang mungkin atau mungkin tidak terafiliasi, terkait dengan, atau disponsori oleh Amazon.

Table of Contents

Apa itu AWS App Mesh?	1
Menambahkan App Mesh ke aplikasi contoh	1
Komponen App Mesh	3
Cara memulai	4
Mengakses App Mesh	4
Memulai	6
App Mesh dan Amazon ECS	6
Skenario	7
Prasyarat	7
Langkah 1: Buat mesh dan layanan virtual	8
Langkah 2: Buat simpul virtual	9
Langkah 3: Buat router virtual dan rute	11
Langkah 4: Tinjau dan buat	13
Langkah 5: Buat sumber daya tambahan	14
Langkah 6: Perbarui layanan	19
Topik lanjutan	35
App Mesh dan Kubernetes	35
Prasyarat	36
Langkah 1: Instal komponen integrasi	37
Langkah 2: Menyebarkan sumber daya App Mesh	43
Langkah 3: Buat atau perbarui layanan	57
Langkah 4: Membersihkan	64
App Mesh dan Amazon EC2	64
Skenario	7
Prasyarat	7
Langkah 1: Buat mesh dan layanan virtual	66
Langkah 2: Buat simpul virtual	67
Langkah 3: Buat router virtual dan rute	11
Langkah 4: Tinjau dan buat	13
Langkah 5: Buat sumber daya tambahan	14
Langkah 6: Perbarui layanan	19
Contoh App Mesh	88
Konsep	89
Jerat	89

Membuat mesh layanan	90
Menghapus mesh	93
Layanan virtual	94
Membuat layanan virtual	94
Menghapus layanan virtual	97
Gerbang virtual	99
Membuat gateway virtual	100
Menyebarkan gateway virtual	105
Menghapus gateway virtual	106
Rute gerbang	107
Node virtual	114
Membuat simpul virtual	115
Menghapus simpul virtual	125
Router virtual	127
Membuat router virtual	128
Menghapus router virtual	130
Rute	132
Utusan	144
Varian gambar utusan	144
Variabel konfigurasi utusan	148
Variabel yang dibutuhkan	149
Variabel opsional	149
Default utusan ditetapkan oleh App Mesh	157
Kebijakan coba lagi rute default	157
Pemutus sirkuit default	158
Memperbarui/memigrasi ke Utusan 1.17	159
Secret Discovery Service dengan SPIRE	159
Perubahan ekspresi reguler	159
Referensi belakang	162
Agen untuk Utusan	163
Observabilitas	165
Pencatatan log	165
Firelens dan Cloudwatch	168
Metrik utusan	168
Contoh metrik aplikasi	171
Mengekspor metrik	175

Pelacakan	184
X-Ray	185
Jaeger	187
Datadog untuk melacak	153
Perkakas	189
CloudFormation	189
AWS CDK	189
Pengontrol App Mesh untuk Kubernetes	190
Terraform	190
Bekerja dengan jerat bersama	191
Memberikan izin untuk berbagi jerat	191
Memberikan izin untuk berbagi mesh	191
Memberikan izin untuk mesh	192
Prasyarat untuk berbagi jerat	193
Layanan terkait	194
Berbagi jala	194
Membatalkan berbagi mesh bersama	195
Mengidentifikasi mesh bersama	196
Tagihan dan pengukuran	196
Kuota contoh	196
Bekerja dengan layanan yang lain	197
Membuat sumber daya App Mesh dengan AWS CloudFormation	197
App Mesh dan CloudFormation template	198
Pelajari lebih lanjut tentang CloudFormation	198
App Mesh di AWS Outposts	198
Prasyarat	199
Batasan	199
Pertimbangan konektivitas jaringan	199
Membuat proxy App Mesh Envoy di Outpost	200
Praktik terbaik	201
Instrumen semua rute dengan percobaan ulang	201
Sesuaikan kecepatan penyebaran	202
Skalakan sebelum skala di	203
Melaksanakan pemeriksaan kesehatan kontainer	203
Optimalkan resolusi DNS	204
Mengamankan Aplikasi	205

Keamanan Lapisan Pengangkutan (TLS)	206
Persyaratan sertifikat	207
Sertifikat otentikasi TLS	208
Bagaimana App Mesh mengonfigurasi Utusan untuk menegosiasikan TLS	210
Verifikasi enkripsi	211
Perpanjangan sertifikat	212
Konfigurasi beban kerja Amazon ECS untuk menggunakan otentikasi TLS AWS App Mesh	213
Konfigurasi beban kerja Kubernetes untuk menggunakan otentikasi TLS dengan AWS App Mesh	213
Autentikasi TLS bersama	214
Sertifikat otentikasi TLS timbal balik	215
Konfigurasi titik akhir mesh	215
Migrasikan layanan ke otentikasi TLS bersama	216
Memverifikasi otentikasi TLS timbal balik	217
Panduan otentikasi TLS timbal balik App Mesh	217
Manajemen identitas dan akses	218
Audiens	219
Mengautentikasi dengan identitas	219
Mengelola akses menggunakan kebijakan	220
Bagaimana AWS App Mesh bekerja dengan IAM	222
Contoh Kebijakan Berbasis Identitas	226
AWS kebijakan terkelola	232
Menggunakan Peran Terkait Layanan	235
Otorisasi Proxy Utusan	238
Pemecahan masalah	244
CloudTrail log	246
Acara manajemen App Mesh di CloudTrail	248
Contoh acara App Mesh	248
Perlindungan data	249
Enkripsi data	250
Validasi kepatuhan	251
Keamanan infrastruktur	252
Antarmuka VPC endpoint (AWS PrivateLink)	252
Ketahanan	254
Pemulihan bencana di AWS App Mesh	255

Konfigurasi dan analisis kerentanan	255
Pemecahan masalah	257
Praktik terbaik	257
Aktifkan antarmuka administrasi proxy Envoy	258
Aktifkan integrasi Envoy DogStats D untuk pembongkaran metrik	258
Aktifkan log akses	259
Aktifkan logging debug Envoy di lingkungan pra-produksi	259
Pantau Konektivitas Proxy Utusan dengan bidang kontrol App Mesh	259
Pengaturan	260
Tidak dapat menarik gambar wadah Utusan	260
Tidak dapat terhubung ke layanan manajemen App Mesh Envoy	261
Utusan terputus dari layanan manajemen App Mesh Envoy dengan teks kesalahan	262
Pemeriksaan kesehatan wadah utusan, pemeriksaan kesiapan, atau penyelidikan keaktifan gagal	265
Pemeriksaan kesehatan dari penyeimbang beban ke titik akhir mesh gagal	265
Gateway virtual tidak menerima lalu lintas di port 1024 atau kurang	266
Konektivitas	267
Tidak dapat menyelesaikan nama DNS untuk layanan virtual	267
Tidak dapat terhubung ke backend layanan virtual	268
Tidak dapat terhubung ke layanan eksternal	270
Tidak dapat terhubung ke server MySQL atau SMTP	270
Tidak dapat terhubung ke layanan yang dimodelkan sebagai node virtual TCP atau router virtual di App Mesh	271
Konektivitas berhasil ke layanan yang tidak terdaftar sebagai backend layanan virtual untuk node virtual	272
Beberapa permintaan gagal dengan kode status HTTP 503 ketika layanan virtual memiliki penyedia node virtual	273
Tidak dapat terhubung ke sistem file Amazon EFS	273
Konektivitas berhasil dilayani, tetapi permintaan yang masuk tidak muncul di log akses, jejak, atau metrik untuk Utusan	274
Menyetel variabelHTTP_PROXY/HTTPS_PROXYenvironment pada level container tidak berfungsi seperti yang diharapkan.	275
Batas waktu permintaan hulu bahkan setelah mengatur batas waktu untuk rute.	275
Utusan merespons dengan permintaan HTTP Bad.	276
Tidak dapat mengonfigurasi batas waktu dengan benar.	277
Penskalaan	277

Konektivitas gagal dan pemeriksaan kesehatan kontainer gagal saat menskalakan melebihi 50 replika untuk gateway virtual node/virtual	278
Permintaan gagal 503 ketika backend layanan virtual secara horizontal keluar atau masuk	278
Kontainer utusan mogok dengan segfault di bawah peningkatan beban	279
Peningkatan sumber daya default tidak tercermin dalam Batas Layanan	279
Aplikasi macet karena sejumlah besar panggilan pemeriksaan kesehatan.	279
Observabilitas	280
Tidak dapat melihat AWS X-Ray jejak untuk aplikasi saya	280
Tidak dapat melihat metrik Utusan untuk aplikasi saya di metrik Amazon CloudWatch	281
Tidak dapat mengonfigurasi aturan pengambilan sampel khusus untuk jejak AWS X-Ray	282
Keamanan	283
Tidak dapat terhubung ke layanan virtual backend dengan kebijakan klien TLS	284
Tidak dapat terhubung ke layanan virtual backend saat aplikasi berasal dari TLS	285
Tidak dapat menegaskan bahwa konektivitas antara proxy Utusan menggunakan TLS	286
Memecahkan Masalah TLS dengan Elastic Load Balancing	287
Kubernetes	288
Sumber daya App Mesh yang dibuat di Kubernetes tidak dapat ditemukan di App Mesh	289
Pod gagal memeriksa kesiapan dan keaktifan setelah envoy sidecar disuntikkan	289
Pod tidak mendaftar atau membatalkan pendaftaran sebagai instance AWS Cloud Map	290
Tidak dapat menentukan lokasi pod untuk resource App Mesh berjalan	291
Tidak dapat menentukan sumber daya App Mesh apa yang dijalankan pod	291
Utusan Klien tidak dapat berkomunikasi dengan Layanan Manajemen Utusan App Mesh dengan dinonaktifkan IMDSv1	292
IRSA tidak berfungsi pada wadah aplikasi saat App Mesh diaktifkan dan Utusan disuntikkan	293
Kuota layanan	294
Riwayat dokumen	296
.....	ccciiii

Apa itu AWS App Mesh?

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk. AWS App Mesh Setelah 30 September 2026, Anda tidak lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

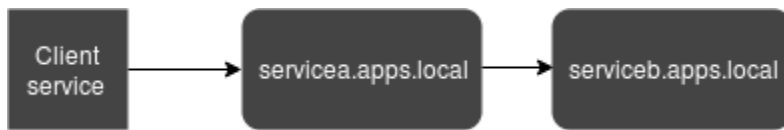
AWS App Mesh adalah mesh layanan yang memudahkan untuk memantau dan mengontrol layanan. Service mesh adalah lapisan infrastruktur yang didedikasikan untuk menangani service-to-service komunikasi, biasanya melalui array proxy jaringan ringan yang digunakan bersama kode aplikasi. App Mesh menstandarisasi cara layanan Anda berkomunikasi, memberi Anda end-to-end visibilitas dan membantu memastikan ketersediaan tinggi untuk aplikasi Anda. App Mesh memberikan visibilitas yang konsisten dan kontrol lalu lintas jaringan untuk setiap layanan dalam aplikasi.

Menambahkan App Mesh ke aplikasi contoh

Important

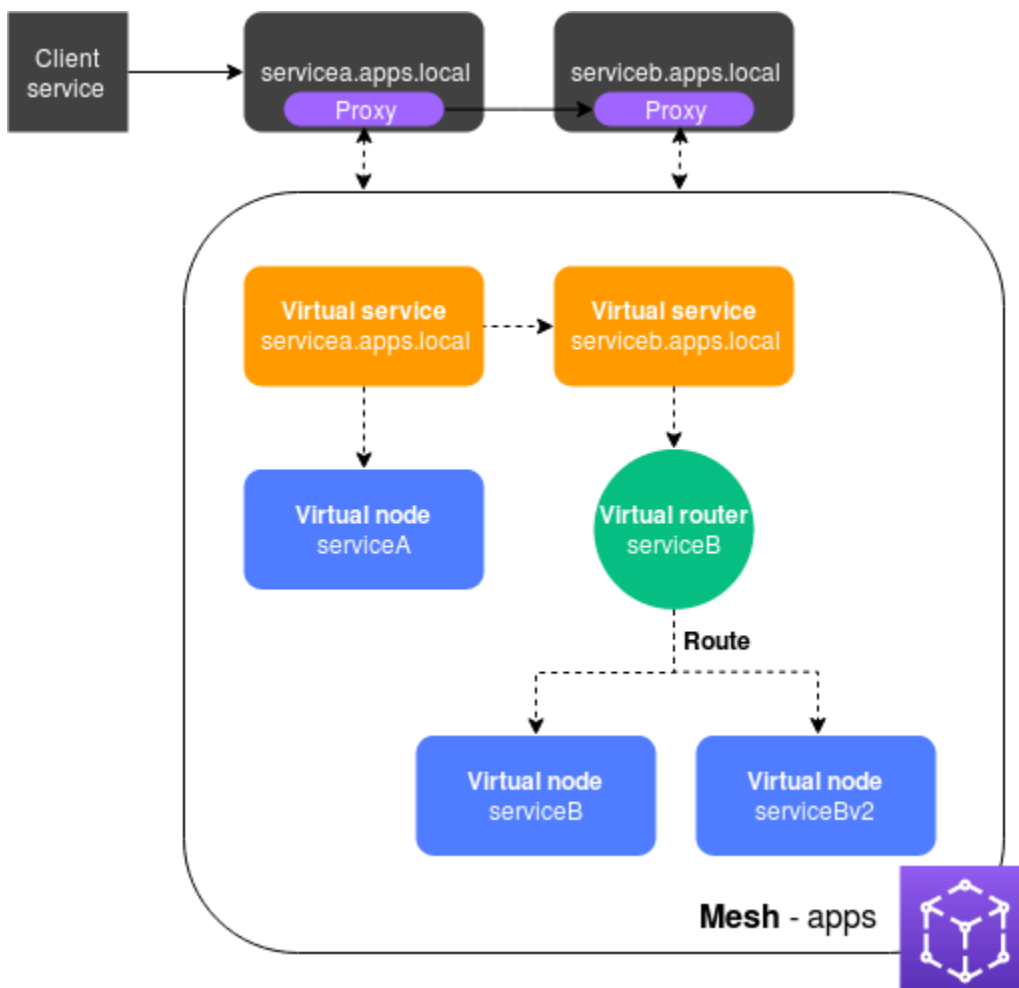
Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk. AWS App Mesh Setelah 30 September 2026, Anda tidak lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Pertimbangkan contoh aplikasi sederhana berikut yang tidak menggunakan App Mesh. Kedua layanan tersebut dapat dijalankan AWS Fargate, Amazon Elastic Container Service (Amazon ECS), Service Elastic Container (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS), Kubernetes di Amazon Elastic Compute Cloud (Amazon) instans, atau di instans Amazon dengan Docker. EC2 EC2



Dalam ilustrasi ini, keduanya `serviceA` dan dapat `serviceB` ditemukan melalui namespace `apps.local`. Katakanlah, misalnya, Anda memutuskan untuk menggunakan versi baru `serviceb.apps.local` bernama `servicebv2.apps.local`. Selanjutnya, Anda ingin mengarahkan persentase lalu lintas dari `servicea.apps.local` ke `serviceb.apps.local` dan persentase ke `servicebv2.apps.local`. Ketika Anda yakin `servicebv2` itu berkinerja baik, Anda ingin mengirim 100 persen lalu lintas ke sana.

App Mesh dapat membantu Anda melakukan ini tanpa mengubah kode aplikasi atau nama layanan terdaftar. Jika Anda menggunakan App Mesh dengan aplikasi contoh ini, maka mesh Anda mungkin terlihat seperti ilustrasi berikut.



Dalam konfigurasi ini, layanan tidak lagi berkomunikasi satu sama lain secara langsung. Sebaliknya, mereka berkomunikasi satu sama lain melalui proxy. Proxy yang digunakan dengan

`servicea.apps.local` layanan membaca konfigurasi App Mesh dan mengirimkan lalu lintas ke `serviceb.apps.local` atau `servicebv2.apps.local` berdasarkan konfigurasi.

Komponen App Mesh

App Mesh terdiri dari komponen-komponen berikut, diilustrasikan dalam contoh sebelumnya:

- **Service mesh** — Service mesh adalah batas logis untuk lalu lintas jaringan antara layanan yang berada di dalamnya. Dalam contoh, mesh diberi nama `apps`, dan berisi semua sumber daya lain untuk mesh. Untuk informasi selengkapnya, lihat [Jala Layanan](#).
- **Layanan virtual** — Layanan virtual adalah abstraksi dari layanan aktual yang disediakan oleh node virtual, secara langsung atau tidak langsung, melalui router virtual. Dalam ilustrasi, dua layanan virtual mewakili dua layanan aktual. Nama-nama layanan virtual adalah nama yang dapat ditemukan dari layanan yang sebenarnya. Ketika layanan virtual dan layanan aktual memiliki nama yang sama, beberapa layanan dapat berkomunikasi satu sama lain menggunakan nama yang sama yang mereka gunakan sebelum App Mesh diimplementasikan. Untuk informasi selengkapnya, lihat [Layanan virtual](#).
- **Node virtual** — Node virtual bertindak sebagai pointer logis ke layanan yang dapat ditemukan, seperti Amazon ECS atau layanan Kubernetes. Untuk setiap layanan virtual, Anda akan memiliki setidaknya satu node virtual. Dalam ilustrasi, layanan `servicea.apps.local` virtual mendapatkan informasi konfigurasi untuk node virtual bernama `serviceA`. Node `serviceA` virtual dikonfigurasi dengan `servicea.apps.local` nama untuk penemuan layanan. Layanan `serviceb.apps.local` virtual dikonfigurasi untuk mengarahkan lalu lintas ke node `serviceBv2` virtual `serviceB` dan virtual melalui router virtual bernama `serviceB`. Untuk informasi selengkapnya, lihat [Node virtual](#).
- **Router dan rute virtual** — Router virtual menangani lalu lintas untuk satu atau lebih layanan virtual dalam mesh Anda. Rute dikaitkan dengan router virtual. Rute ini digunakan untuk mencocokkan permintaan untuk router virtual dan untuk mendistribusikan lalu lintas ke node virtual yang terkait. Dalam ilustrasi sebelumnya, router `serviceB` virtual memiliki rute yang mengarahkan persentase lalu lintas ke node `serviceB` virtual, dan persentase lalu lintas ke node `serviceBv2` virtual. Anda dapat mengatur persentase lalu lintas yang diarahkan ke node virtual tertentu dan mengubahnya dari waktu ke waktu. Anda dapat merutekan lalu lintas berdasarkan kriteria seperti header HTTP, jalur URL, atau nama layanan dan metode gRPC. Anda dapat mengonfigurasi kebijakan coba lagi untuk mencoba kembali koneksi jika ada kesalahan dalam respons. Misalnya, dalam ilustrasi, kebijakan coba lagi untuk rute dapat menentukan bahwa koneksi ke `serviceb.apps.local` dicoba ulang lima kali, dengan sepuluh detik antara upaya coba lagi, jika `serviceb.apps.local`

mengembalikan jenis kesalahan tertentu. Untuk informasi selengkapnya, lihat [Router virtual](#) dan [Rute](#).

- Proxy — Anda mengonfigurasi layanan Anda untuk menggunakan proxy setelah Anda membuat mesh dan sumber dayanya. Proxy membaca konfigurasi App Mesh dan mengarahkan lalu lintas dengan tepat. Dalam ilustrasi, semua komunikasi dari `servicea.apps.local` hingga `serviceb.apps.local` melewati proxy yang digunakan dengan setiap layanan. Layanan berkomunikasi satu sama lain menggunakan nama penemuan layanan yang sama yang mereka gunakan sebelum memperkenalkan App Mesh. Karena proxy membaca konfigurasi App Mesh, Anda dapat mengontrol cara kedua layanan berkomunikasi satu sama lain. Bila ingin mengubah konfigurasi App Mesh, Anda tidak perlu mengubah atau menerapkan ulang layanan itu sendiri atau proxy. Untuk informasi selengkapnya, lihat [Gambar utusan](#).

Cara memulai

Untuk menggunakan App Mesh, Anda harus memiliki layanan yang sudah berjalan, Amazon ECS, AWS Fargate, Amazon EKS, Kubernetes di Amazon EC2, atau Amazon EC2 dengan Docker.

Untuk memulai App Mesh, lihat salah satu panduan berikut:

- [Memulai dengan App Mesh dan Amazon ECS](#)
- [Memulai App Mesh dan Kubernetes](#)
- [Memulai dengan App Mesh dan Amazon EC2](#)

Mengakses App Mesh

Anda dapat bekerja dengan App Mesh dengan cara berikut:

Konsol Manajemen AWS

Konsol adalah antarmuka berbasis browser yang dapat Anda gunakan untuk mengelola sumber daya App Mesh. Anda dapat membuka konsol App Mesh di <https://console.aws.amazon.com/appmesh/>.

AWS CLI

Menyediakan perintah untuk serangkaian AWS produk yang luas, dan didukung di Windows, Mac, dan Linux. Untuk mulai, lihat [Panduan Pengguna AWS Command Line Interface](#). Untuk informasi selengkapnya tentang perintah untuk App Mesh, lihat [appmesh](#) di [Referensi AWS CLI Perintah](#).

AWS Tools for Windows PowerShell

Memberikan perintah untuk serangkaian AWS produk yang luas bagi mereka yang membuat skrip di PowerShell lingkungan. Untuk memulai, lihat [Panduan Pengguna Alat AWS untuk PowerShell](#). Untuk informasi selengkapnya tentang cmdlet untuk App Mesh, lihat [App Mesh](#) di [AWS Alat untuk Referensi PowerShell Cmdlet](#).

AWS CloudFormation

Memungkinkan Anda membuat template yang menjelaskan semua sumber AWS daya yang Anda inginkan. Menggunakan template, CloudFormation ketentuan dan mengkonfigurasi sumber daya untuk Anda. Untuk mulai, lihat [Panduan Pengguna AWS CloudFormation](#). Untuk informasi selengkapnya tentang jenis resource App Mesh, lihat [Referensi Jenis Sumber Daya App Mesh di Referensi AWS CloudFormation Templat](#).

AWS SDKs

Kami juga menyediakan SDKs yang memungkinkan Anda mengakses App Mesh dari berbagai bahasa pemrograman. SDKs Secara otomatis mengurus tugas-tugas seperti:

- Secara kriptografi menandatangani permintaan layanan Anda
- Mencoba kembali permintaan
- Menangani respons kesalahan

Untuk informasi selengkapnya tentang tersedia SDKs, lihat [Alat untuk Amazon Web Services](#).

Untuk informasi selengkapnya tentang App Mesh APIs, lihat [Referensi AWS App Mesh API](#).

Memulai dengan App Mesh

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk. AWS App Mesh Setelah 30 September 2026, Anda tidak akan lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Anda dapat menggunakan App Mesh dengan aplikasi yang Anda terapkan ke Amazon ECS, Kubernetes (yang Anda terapkan ke instans Amazon EC2 Anda sendiri atau berjalan di Amazon EKS), dan Amazon EC2. Untuk memulai App Mesh, pilih salah satu layanan yang Anda miliki aplikasi yang digunakan untuk yang ingin Anda gunakan dengan App Mesh. Anda selalu dapat mengaktifkan aplikasi di layanan lain untuk juga bekerja dengan App Mesh setelah Anda menyelesaikan salah satu panduan Memulai.

Topik

- [Memulai dengan AWS App Mesh dan Amazon ECS](#)
- [Memulai dengan AWS App Mesh dan Kubernetes](#)
- [Memulai dengan AWS App Mesh dan Amazon EC2](#)
- [Contoh App Mesh](#)

Memulai dengan AWS App Mesh dan Amazon ECS

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk. AWS App Mesh Setelah 30 September 2026, Anda tidak akan lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Topik ini membantu Anda menggunakan AWS App Mesh layanan aktual yang berjalan di Amazon ECS. Tutorial ini mencakup fitur dasar dari beberapa jenis sumber daya App Mesh.

Skenario

Untuk mengilustrasikan cara menggunakan App Mesh, asumsikan bahwa Anda memiliki aplikasi dengan karakteristik sebagai berikut:

- Terdiri dari dua layanan bernama `serviceA` dan `serviceB`.
- Kedua layanan terdaftar ke namespace bernama `apps.local`
- `ServiceA` berkomunikasi dengan `serviceB` lebih dari HTTP/2, port 80.
- Anda telah menerapkan versi 2 `serviceB` dan mendaftarkannya dengan nama `serviceBv2` di `apps.local` namespace.

Anda memiliki persyaratan berikut:

- Anda ingin mengirim 75 persen lalu lintas dari `serviceA` ke `serviceB` dan 25 persen lalu lintas ke yang `serviceBv2` pertama. Dengan hanya mengirim 25 persen ke `serviceBv2`, Anda dapat memvalidasi bahwa itu bebas bug sebelum Anda mengirim 100 persen lalu lintas dari `serviceA`.
- Anda ingin dapat dengan mudah menyesuaikan bobot lalu lintas sehingga 100 persen lalu lintas masuk `serviceBv2` setelah terbukti dapat diandalkan. Setelah semua lalu lintas dikirim ke `serviceBv2`, Anda ingin berhenti `serviceB`.
- Anda tidak ingin harus mengubah kode aplikasi atau pendaftaran penemuan layanan yang ada untuk layanan Anda yang sebenarnya untuk memenuhi persyaratan sebelumnya.

Untuk memenuhi kebutuhan Anda, Anda memutuskan untuk membuat mesh layanan App Mesh dengan layanan virtual, node virtual, router virtual, dan rute. Setelah menerapkan mesh Anda, Anda memperbarui layanan Anda untuk menggunakan proxy Envoy. Setelah diperbarui, layanan Anda berkomunikasi satu sama lain melalui proxy Utusan daripada langsung satu sama lain.

Prasyarat

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk. AWS App Mesh Setelah 30 September 2026, Anda tidak akan lagi dapat

mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

- Pemahaman yang ada tentang konsep App Mesh. Untuk informasi selengkapnya, lihat [Apa itu AWS App Mesh?](#).
- Pemahaman yang ada tentang ECSs konsep Amazon. Untuk informasi selengkapnya, lihat [Apa itu Amazon ECS](#) di Panduan Pengembang Layanan Kontainer Elastis Amazon.
- App Mesh mendukung layanan Linux yang terdaftar dengan DNS, AWS Cloud Map, atau keduanya. Untuk menggunakan panduan memulai ini, kami sarankan Anda memiliki tiga layanan yang sudah ada yang terdaftar di DNS. Prosedur dalam topik ini mengasumsikan bahwa layanan yang ada diberi nama `serviceA`, `serviceB`, `serviceBv2` dan bahwa semua layanan dapat ditemukan melalui namespace bernama `apps.local`

Anda dapat membuat mesh layanan dan sumber dayanya bahkan jika layanan tidak ada, tetapi Anda tidak dapat menggunakan mesh sampai Anda telah menerapkan layanan yang sebenarnya. Untuk informasi selengkapnya tentang penemuan layanan di Amazon ECS, lihat [Penemuan Layanan](#). Untuk membuat layanan Amazon ECS dengan penemuan layanan, lihat [Tutorial: Membuat Layanan Menggunakan Penemuan Layanan](#). Jika Anda belum menjalankan layanan, Anda dapat [membuat layanan Amazon ECS dengan penemuan layanan](#).

Langkah 1: Buat mesh dan layanan virtual

Mesh layanan adalah batas logis untuk lalu lintas jaringan antara layanan yang berada di dalamnya. Untuk informasi selengkapnya, lihat [Jala Layanan](#). Layanan virtual adalah abstraksi dari layanan yang sebenarnya. Untuk informasi selengkapnya, lihat [Layanan virtual](#).

Buat sumber daya berikut:

- Sebuah mesh bernama `apps`, karena semua layanan dalam skenario terdaftar ke `apps.local` namespace.
- Layanan virtual bernama `serviceb.apps.local`, karena layanan virtual mewakili layanan yang dapat ditemukan dengan nama itu, dan Anda tidak ingin mengubah kode Anda untuk merujuk nama lain. Layanan virtual bernama `servicea.apps.local` ditambahkan pada langkah selanjutnya.

Anda dapat menggunakan Konsol Manajemen AWS atau AWS CLI versi 1.18.116 atau lebih tinggi atau 2.0.38 atau lebih tinggi untuk menyelesaikan langkah-langkah berikut. Jika menggunakan AWS CLI, gunakan `aws --version` perintah untuk memeriksa AWS CLI versi yang Anda instal. [Jika Anda tidak memiliki versi 1.18.116 atau lebih tinggi atau 2.0.38 atau lebih tinggi diinstal, maka Anda harus menginstal atau memperbarui. AWS CLI](#) Pilih tab untuk alat yang ingin Anda gunakan.

Konsol Manajemen AWS

1. Buka wizard yang dijalankan pertama kali konsol App Mesh saat memulai <https://console.aws.amazon.com/appmesh/>.
2. Untuk nama Mesh, masukkan **apps**.
3. Untuk nama layanan Virtual, masukkan **serviceb.apps.local**.
4. Untuk melanjutkan, pilih Berikutnya.

AWS CLI

1. Buat mesh dengan [create-mesh](#) perintah.

```
aws appmesh create-mesh --mesh-name apps
```

2. Buat layanan virtual dengan [create-virtual-service](#) perintah.

```
aws appmesh create-virtual-service --mesh-name apps --virtual-service-name serviceb.apps.local --spec {}
```

Langkah 2: Buat simpul virtual

Node virtual bertindak sebagai pointer logis ke layanan yang sebenarnya. Untuk informasi selengkapnya, lihat [Node virtual](#).

Buat node virtual bernama `serviceB`, karena salah satu node virtual mewakili layanan yang sebenarnya bernama `serviceB`. Layanan aktual yang diwakili oleh node virtual dapat ditemukan melalui DNS dengan nama host dari `serviceb.apps.local` Sebagai alternatif, Anda dapat menemukan layanan aktual menggunakan AWS Cloud Map Node virtual akan mendengarkan lalu lintas menggunakan protokol HTTP/2 pada port 80. Protokol lain juga didukung, seperti halnya pemeriksaan kesehatan. Anda akan membuat node virtual untuk `serviceA` dan `serviceBv2` di langkah selanjutnya.

Konsol Manajemen AWS

1. Untuk nama simpul Virtual, masukkan **serviceB**.
2. Untuk metode Penemuan layanan, pilih DNS dan masukkan **serviceb.apps.local** untuk nama host DNS.
3. Di bawah konfigurasi Listener, pilih http2 untuk Protokol dan masukkan **80** untuk Port.
4. Untuk melanjutkan, pilih Berikutnya.

AWS CLI

1. Buat file bernama `create-virtual-node-serviceb.json` dengan konten berikut:

```
{
  "meshName": "apps",
  "spec": {
    "listeners": [
      {
        "portMapping": {
          "port": 80,
          "protocol": "http2"
        }
      }
    ],
    "serviceDiscovery": {
      "dns": {
        "hostname": "serviceb.apps.local"
      }
    }
  },
  "virtualNodeName": "serviceB"
}
```

2. Buat node virtual dengan [create-virtual-node](#) perintah menggunakan file JSON sebagai input.

```
aws appmesh create-virtual-node --cli-input-json file://create-virtual-node-  
serviceb.json
```

Langkah 3: Buat router virtual dan rute

Router virtual merutekan lalu lintas untuk satu atau lebih layanan virtual di dalam mesh Anda. Untuk informasi selengkapnya, lihat [Router virtual](#) dan [Rute](#).

Buat sumber daya berikut:

- Router virtual bernam `serviceB`, karena layanan `serviceB.apps.local` virtual tidak memulai komunikasi keluar dengan layanan lain. Ingatlah bahwa layanan virtual yang Anda buat sebelumnya adalah abstraksi dari `serviceB.apps.local` layanan Anda yang sebenarnya. Layanan virtual mengirimkan lalu lintas ke router virtual. Router virtual mendengarkan lalu lintas menggunakan protokol HTTP/2 pada port 80. Protokol lain juga didukung.
- Sebuah rute bernam `serviceB`. Ini merutekan 100 persen lalu lintasnya ke node `serviceB` virtual. Bobotnya ada di langkah selanjutnya setelah Anda menambahkan simpul `serviceBv2` virtual. Meskipun tidak tercakup dalam panduan ini, Anda dapat menambahkan kriteria filter tambahan untuk rute dan menambahkan kebijakan coba lagi untuk menyebabkan proxy Envoy melakukan beberapa upaya untuk mengirim lalu lintas ke node virtual ketika mengalami masalah komunikasi.

Konsol Manajemen AWS

1. Untuk nama router Virtual, masukkan **`serviceB`**.
2. Di bawah konfigurasi Listener, pilih `http2` untuk Protokol dan tentukan **`80`** untuk Port.
3. Untuk nama Rute, masukkan **`serviceB`**.
4. Untuk jenis Rute, pilih `http2`.
5. Untuk nama simpul Virtual di bawah konfigurasi Target, pilih `serviceB` dan masukkan **`100`** untuk Berat.
6. Di bawah konfigurasi Match, pilih Metode.
7. Untuk melanjutkan, pilih Berikutnya.

AWS CLI

1. Buat router virtual.
 - a. Buat file bernama `create-virtual-router.json` dengan konten berikut:

```
{
  "meshName": "apps",
  "spec": {
    "listeners": [
      {
        "portMapping": {
          "port": 80,
          "protocol": "http2"
        }
      }
    ]
  },
  "virtualRouterName": "serviceB"
}
```

- b. Buat router virtual dengan [create-virtual-router](#) perintah menggunakan file JSON sebagai input.

```
aws appmesh create-virtual-router --cli-input-json file://create-virtual-router.json
```

2. Buat rute.

- a. Buat file bernama `create-route.json` dengan konten berikut:

```
{
  "meshName" : "apps",
  "routeName" : "serviceB",
  "spec" : {
    "httpRoute" : {
      "action" : {
        "weightedTargets" : [
          {
            "virtualNode" : "serviceB",
            "weight" : 100
          }
        ]
      },
      "match" : {
        "prefix" : "/"
      }
    }
  }
}
```

```
  },  
  "virtualRouterName" : "serviceB"  
}
```

- b. Buat rute dengan perintah [create-route](#) menggunakan file JSON sebagai input.

```
aws appmesh create-route --cli-input-json file://create-route.json
```

Langkah 4: Tinjau dan buat

Tinjau pengaturan terhadap instruksi sebelumnya.

Konsol Manajemen AWS

Pilih Edit jika Anda perlu membuat perubahan di bagian mana pun. Setelah Anda puas dengan pengaturan, pilih Buat mesh.

Layar Status menunjukkan kepada Anda semua sumber daya mesh yang dibuat. Anda dapat melihat sumber daya yang dibuat di konsol dengan memilih View mesh.

AWS CLI

Tinjau pengaturan mesh yang Anda buat dengan [perintah describe-mesh](#).

```
aws appmesh describe-mesh --mesh-name apps
```

Tinjau pengaturan layanan virtual yang Anda buat dengan [describe-virtual-service](#) perintah.

```
aws appmesh describe-virtual-service --mesh-name apps --virtual-service-name  
serviceb.apps.local
```

Tinjau pengaturan node virtual yang Anda buat dengan [describe-virtual-node](#) perintah.

```
aws appmesh describe-virtual-node --mesh-name apps --virtual-node-name serviceB
```

Tinjau pengaturan router virtual yang Anda buat dengan [describe-virtual-router](#) perintah.

```
aws appmesh describe-virtual-router --mesh-name apps --virtual-router-name serviceB
```

Tinjau pengaturan rute yang Anda buat dengan [perintah deskripsi-rute](#).

```
aws appmesh describe-route --mesh-name apps \  
  --virtual-router-name serviceB --route-name serviceB
```

Langkah 5: Buat sumber daya tambahan

Untuk menyelesaikan skenario, Anda perlu:

- Buat satu node virtual bernama `serviceBv2` dan yang lain bernama `serviceA`. Kedua node virtual mendengarkan permintaan melalui HTTP/2 port 80. Untuk node `serviceA` virtual, konfigurasi backend dari `serviceb.apps.local`. Semua lalu lintas keluar dari node `serviceA` virtual dikirim ke layanan virtual bernama `serviceb.apps.local`. Meskipun tidak tercakup dalam panduan ini, Anda juga dapat menentukan jalur file untuk menulis log akses untuk node virtual.
- Buat satu layanan virtual tambahan bernama `servicea.apps.local`, yang mengirimkan semua lalu lintas langsung ke node `serviceA` virtual.
- Perbarui `serviceB` rute yang Anda buat pada langkah sebelumnya untuk mengirim 75 persen lalu lintasnya ke node `serviceB` virtual dan 25 persen lalu lintasnya ke node `serviceBv2` virtual. Seiring waktu, Anda dapat terus memodifikasi bobot hingga `serviceBv2` menerima 100 persen dari lalu lintas. Setelah semua lalu lintas dikirim ke `serviceBv2`, Anda dapat mematikan dan menghentikan node `serviceB` virtual dan layanan aktual. Saat Anda mengubah bobot, kode Anda tidak memerlukan modifikasi apa pun, karena nama layanan `serviceb.apps.local` virtual dan aktual tidak berubah. Ingatlah bahwa layanan `serviceb.apps.local` virtual mengirimkan lalu lintas ke router virtual, yang merutekan lalu lintas ke node virtual. Nama penemuan layanan untuk node virtual dapat diubah kapan saja.

Konsol Manajemen AWS

1. Di panel navigasi kiri, pilih Meshes.
2. Pilih apps mesh yang Anda buat pada langkah sebelumnya.
3. Di panel navigasi kiri, pilih Virtual nodes.
4. Pilih Buat simpul virtual.
5. Untuk nama simpul Virtual, masukkan `serviceBv2`, untuk metode penemuan Layanan, pilih DNS, dan untuk nama host DNS, masukkan `servicebv2.apps.local`
6. Untuk konfigurasi Listener, pilih http2 untuk Protokol dan masukkan `80` untuk Port.

7. Pilih Buat simpul virtual.
8. Pilih Buat simpul virtual lagi. Masukkan **serviceA** untuk nama simpul Virtual. Untuk metode Penemuan layanan, pilih DNS, dan untuk nama host DNS, masukkan **servicea.apps.local**
9. Untuk Masukkan nama layanan virtual di bawah Backend baru, masukkan **serviceb.apps.local**
10. Di bawah konfigurasi Listener, pilih http2 untuk Protokol, masukkan **80** untuk Port, dan kemudian pilih Buat simpul virtual.
11. Di panel navigasi kiri, pilih Router virtual dan kemudian pilih router `serviceB` virtual dari daftar.
12. Di bawah Rute, pilih rute bernama **ServiceB** yang Anda buat pada langkah sebelumnya, dan pilih Edit.
13. Di bawah Target, nama node Virtual, ubah nilai Weight `serviceB` untuk menjadi **75**.
14. Pilih Tambah target, pilih `serviceBv2` dari daftar dropdown, dan atur nilai Weight ke. **25**
15. Pilih Simpan.
16. Di panel navigasi kiri, pilih Layanan virtual dan kemudian pilih Buat layanan virtual.
17. Masukkan **servicea.apps.local** nama layanan Virtual, pilih Virtual node for Provider, pilih `serviceA` untuk Virtual node, dan kemudian pilih Create Virtual Service.

AWS CLI

1. Buat node `serviceBv2` virtual.
 - a. Buat file bernama `create-virtual-node-servicebv2.json` dengan konten berikut:

```
{
  "meshName": "apps",
  "spec": {
    "listeners": [
      {
        "portMapping": {
          "port": 80,
          "protocol": "http2"
        }
      }
    ]
  },
}
```

```
    "serviceDiscovery": {
      "dns": {
        "hostname": "serviceBv2.apps.local"
      }
    },
    "virtualNodeName": "serviceBv2"
  }
}
```

- b. Buat node virtual.

```
aws appmesh create-virtual-node --cli-input-json file://create-virtual-node-  
servicebv2.json
```

2. Buat node serviceA virtual.

- a. Buat file bernama `create-virtual-node-servicea.json` dengan konten berikut:

```
{
  "meshName" : "apps",
  "spec" : {
    "backends" : [
      {
        "virtualService" : {
          "virtualServiceName" : "serviceb.apps.local"
        }
      }
    ],
    "listeners" : [
      {
        "portMapping" : {
          "port" : 80,
          "protocol" : "http2"
        }
      }
    ],
    "serviceDiscovery" : {
      "dns" : {
        "hostname" : "servicea.apps.local"
      }
    }
  },
  "virtualNodeName" : "serviceA"
}
```

```
}  
}
```

- b. Buat node virtual.

```
aws appmesh create-virtual-node --cli-input-json file://create-virtual-node-servicea.json
```

3. Perbarui layanan `serviceb.apps.local` virtual yang Anda buat pada langkah sebelumnya untuk mengirim lalu lintas ke router `serviceB` virtual. Ketika layanan virtual awalnya dibuat, itu tidak mengirim lalu lintas ke mana pun, karena router `serviceB` virtual belum dibuat.

- a. Buat file bernama `update-virtual-service.json` dengan konten berikut:

```
{  
  "meshName" : "apps",  
  "spec" : {  
    "provider" : {  
      "virtualRouter" : {  
        "virtualRouterName" : "serviceB"  
      }  
    }  
  },  
  "virtualServiceName" : "serviceb.apps.local"  
}
```

- b. Perbarui layanan virtual dengan [update-virtual-service](#) perintah.

```
aws appmesh update-virtual-service --cli-input-json file://update-virtual-service.json
```

4. Perbarui `serviceB` rute yang Anda buat di langkah sebelumnya.

- a. Buat file bernama `update-route.json` dengan konten berikut:

```
{  
  "meshName" : "apps",  
  "routeName" : "serviceB",  
  "spec" : {  
    "http2Route" : {  
      "action" : {  
        "weightedTargets" : [  
          {  
            "virtualNode" : "serviceB",  

```

```

        "weight" : 75
      },
      {
        "virtualNode" : "serviceBv2",
        "weight" : 25
      }
    ]
  },
  "match" : {
    "prefix" : "/"
  }
}
},
"virtualRouterName" : "serviceB"
}

```

- b. Perbarui rute dengan perintah [update-route](#).

```
aws appmesh update-route --cli-input-json file://update-route.json
```

5. Buat layanan serviceA virtual.

- a. Buat file bernama `create-virtual-servicea.json` dengan konten berikut:

```

{
  "meshName" : "apps",
  "spec" : {
    "provider" : {
      "virtualNode" : {
        "virtualNodeName" : "serviceA"
      }
    }
  },
  "virtualServiceName" : "servicea.apps.local"
}

```

- b. Buat layanan virtual.

```
aws appmesh create-virtual-service --cli-input-json file://create-virtual-servicea.json
```

Ringkasan mesh

Sebelum Anda membuat mesh layanan, Anda memiliki tiga layanan aktual bernama `servicea.apps.local`, `serviceb.apps.local`, dan `servicebv2.apps.local`. Selain layanan yang sebenarnya, Anda sekarang memiliki mesh layanan yang berisi sumber daya berikut yang mewakili layanan sebenarnya:

- Dua layanan virtual. Proxy mengirimkan semua lalu lintas dari layanan `servicea.apps.local` virtual ke layanan `serviceb.apps.local` virtual melalui router virtual.
- Tiga node virtual bernama `serviceA`, `serviceB`, dan `serviceBv2`. Proxy Envoy menggunakan informasi penemuan layanan yang dikonfigurasi untuk node virtual untuk mencari alamat IP dari layanan yang sebenarnya.
- Satu router virtual dengan satu rute yang menginstruksikan proxy Utusan untuk merutekan 75 persen lalu lintas masuk ke node `serviceB` virtual dan 25 persen lalu lintas ke node virtual `serviceBv2`.

Langkah 6: Perbarui layanan

Setelah membuat mesh Anda, Anda harus menyelesaikan tugas-tugas berikut:

- Otorisasi proxy Envoy yang Anda terapkan dengan setiap tugas Amazon ECS untuk membaca konfigurasi satu atau beberapa node virtual. Untuk informasi selengkapnya tentang cara mengotorisasi proxy, lihat [Otorisasi proxy](#).
- Perbarui setiap definisi tugas Amazon ECS yang ada untuk menggunakan proxy Envoy.

Kredensial

Container Envoy memerlukan AWS Identity and Access Management kredensial untuk menandatangani permintaan yang dikirim ke layanan App Mesh. [Untuk tugas Amazon ECS yang diterapkan dengan jenis peluncuran Amazon EC2, kredensialnya dapat berasal dari peran instans atau dari peran IAM tugas.](#) Tugas Amazon ECS yang diterapkan dengan Fargate di wadah Linux tidak memiliki akses ke server metadata Amazon EC2 yang menyediakan kredensial profil IAM instans. Untuk menyediakan kredensialnya, Anda harus melampirkan peran tugas IAM ke tugas apa pun yang digunakan dengan Fargate pada jenis container Linux.

Jika tugas diterapkan dengan jenis peluncuran Amazon EC2 dan akses diblokir ke server metadata Amazon EC2, seperti yang dijelaskan dalam anotasi Penting dalam Peran IAM [untuk Tugas, maka peran IAM](#) tugas juga harus dilampirkan ke tugas. Peran yang Anda tetapkan ke instans atau tugas

harus memiliki kebijakan IAM yang dilampirkan padanya seperti yang dijelaskan dalam otorisasi [Proxy](#).

Untuk memperbarui definisi tugas Anda menggunakan AWS CLI

Anda menggunakan AWS CLI perintah [register-task-definition](#) Amazon ECS. Contoh definisi tugas di bawah ini menunjukkan cara mengonfigurasi App Mesh untuk layanan Anda.

Note

Mengonfigurasi App Mesh untuk Amazon ECS melalui konsol tidak tersedia.

Definisi tugas json

Konfigurasi proxy

Untuk mengonfigurasi layanan Amazon ECS agar menggunakan App Mesh, definisi tugas layanan Anda harus memiliki bagian konfigurasi proxy berikut. Atur konfigurasi proxy type ke APPMESH dan containerName keenvoy. Tetapkan nilai properti berikut sesuai.

IgnoredUID

Proxy Envoy tidak merutekan lalu lintas dari proses yang menggunakan ID pengguna ini. Anda dapat memilih ID pengguna apa pun yang Anda inginkan untuk nilai properti ini, tetapi ID ini harus sama dengan user ID untuk wadah Utusan dalam definisi tugas Anda. Pencocokan ini memungkinkan Utusan untuk mengabaikan lalu lintasnya sendiri tanpa menggunakan proxy. Contoh kami digunakan **1337** untuk tujuan sejarah.

ProxyIngressPort

Ini adalah port masuk untuk wadah proxy Envoy. Tetapkan nilai ini ke15000.

ProxyEgressPort

Ini adalah port keluar untuk wadah proxy Envoy. Tetapkan nilai ini ke15001.

AppPorts

Tentukan port masuk yang didengarkan oleh wadah aplikasi Anda. Dalam contoh ini, wadah aplikasi mendengarkan pada port**9080**. Port yang Anda tentukan harus cocok dengan port yang dikonfigurasi pada pendengar simpul virtual.

EgressIgnoredIPs

Utusan tidak mem-proxy lalu lintas ke alamat IP ini. Tetapkan nilai ini ke `169.254.170.2,169.254.169.254`, yang mengabaikan server metadata Amazon EC2 dan titik akhir metadata tugas Amazon ECS. Titik akhir metadata menyediakan peran IAM untuk kredensial tugas. Anda dapat menambahkan alamat tambahan.

EgressIgnoredPorts

Anda dapat menambahkan daftar port yang dipisahkan koma. Utusan tidak mem-proxy lalu lintas ke port ini. Bahkan jika Anda tidak mencantumkan port, port 22 diabaikan.

Note

Jumlah maksimum port keluar yang dapat diabaikan adalah 15.

```
"proxyConfiguration": {
  "type": "APPMESH",
  "containerName": "envoy",
  "properties": [{
    "name": "IgnoredUID",
    "value": "1337"
  },
  {
    "name": "ProxyIngressPort",
    "value": "15000"
  },
  {
    "name": "ProxyEgressPort",
    "value": "15001"
  },
  {
    "name": "AppPorts",
    "value": "9080"
  },
  {
    "name": "EgressIgnoredIPs",
    "value": "169.254.170.2,169.254.169.254"
  },
  {
    "name": "EgressIgnoredPorts",
```

```

    "value": "22"
  }
]
}

```

Ketergantungan Utusan wadah aplikasi

Wadah aplikasi dalam definisi tugas Anda harus menunggu proxy Utusan untuk bootstrap dan mulai sebelum mereka dapat memulai. Untuk memastikan ini terjadi, Anda menetapkan `dependsOn` bagian di setiap definisi wadah aplikasi untuk menunggu kontainer Utusan melaporkan sebagai `HEALTHY`. Kode berikut menunjukkan contoh definisi wadah aplikasi dengan ketergantungan ini. Semua properti dalam contoh berikut diperlukan. Beberapa nilai properti juga diperlukan, tetapi beberapa di antaranya *replaceable*.

```

{
  "name": "appName",
  "image": "appImage",
  "portMappings": [{
    "containerPort": 9080,
    "hostPort": 9080,
    "protocol": "tcp"
  }],
  "essential": true,
  "dependsOn": [{
    "containerName": "envoy",
    "condition": "HEALTHY"
  }]
}

```

Definisi wadah utusan

Definisi tugas Amazon ECS Anda harus berisi gambar wadah App Mesh Envoy.

Semua Wilayah yang [didukung](#) dapat diganti *Region-code* dengan Wilayah apa pun selain `me-south-1`, `ap-east-1`, `ap-southeast-3`, `eu-south-1`, `il-central-1`, dan `af-south-1`.

Standar

```
840364872350.dkr.ecr.region-code.amazonaws.com/aws-appmesh-envoy:v1.34.13.0-prod
```

Sesuai FIPS

```
840364872350.dkr.ecr.region-code.amazonaws.com/aws-appmesh-envoy:v1.34.13.0-prod-  
fips
```

me-south-1

Standar

```
772975370895.dkr.ecr.me-south-1.amazonaws.com/aws-appmesh-envoy:v1.34.13.0-prod
```

ap-east-1

Standar

```
856666278305.dkr.ecr.ap-east-1.amazonaws.com/aws-appmesh-envoy:v1.34.13.0-prod
```

ap-southeast-3

Standar

```
909464085924.dkr.ecr.ap-southeast-3.amazonaws.com/aws-appmesh-envoy:v1.34.13.0-prod
```

eu-south-1

Standar

```
422531588944.dkr.ecr.eu-south-1.amazonaws.com/aws-appmesh-envoy:v1.34.13.0-prod
```

il-central-1

Standar

```
564877687649.dkr.ecr.il-central-1.amazonaws.com/aws-appmesh-envoy:v1.34.13.0-prod
```

af-south-1

Standar

```
924023996002.dkr.ecr.af-south-1.amazonaws.com/aws-appmesh-envoy:v1.34.13.0-prod
```

Public repository

Standar

```
public.ecr.aws/appmesh/aws-appmesh-envoy:v1.34.13.0-prod
```

Sesuai FIPS

```
public.ecr.aws/appmesh/aws-appmesh-envoy:v1.34.13.0-prod-fips
```

Important

Hanya versi v1.9.0.0-prod atau yang lebih baru yang didukung untuk digunakan dengan App Mesh.

Anda harus menggunakan image container App Mesh Envoy hingga tim project Envoy menggabungkan perubahan yang mendukung App Mesh. Untuk detail tambahan, lihat [masalah GitHub peta jalan](#).

Semua properti dalam contoh berikut diperlukan. Beberapa nilai properti juga diperlukan, tetapi beberapa di antaranya *replaceable*.

Note

- Definisi wadah Utusan harus ditandai sebagai `essential`
- Kami merekomendasikan mengalokasikan unit 512 CPU dan setidaknya 64 MiB memori ke wadah Envoy. Di Fargate, yang terendah yang dapat Anda atur adalah memori 1024 MiB.
- Nama node virtual untuk layanan Amazon ECS harus disetel ke nilai `APP_MESH_RESOURCE_ARN` properti. Properti ini memerlukan versi `1.15.0` atau yang lebih baru dari gambar Utusan. Untuk informasi selengkapnya, lihat [Utusan](#).
- Nilai untuk `user` pengaturan harus sesuai dengan `IgnoredUID` nilai dari konfigurasi proxy definisi tugas. Dalam contoh ini, kami menggunakan `1337`.
- Pemeriksaan kesehatan yang ditunjukkan di sini menunggu wadah Utusan untuk melakukan bootstrap dengan benar sebelum melaporkan ke Amazon ECS bahwa wadah Utusan sehat dan siap untuk wadah aplikasi dimulai.
- Secara default, App Mesh menggunakan nama sumber daya yang Anda tentukan `APP_MESH_RESOURCE_ARN` saat Envoy merujuk dirinya sendiri dalam metrik dan

jejak. Anda dapat menimpa perilaku ini dengan mengatur variabel lingkungan APPMESH_RESOURCE_CLUSTER dengan nama Anda sendiri. Properti ini memerlukan versi 1.15.0 atau yang lebih baru dari gambar Utusan. Untuk informasi selengkapnya, lihat [Utusan](#).

Kode berikut menunjukkan contoh definisi kontainer Utusan.

```
{
  "name": "envoy",
  "image": "840364872350.dkr.ecr.us-west-2.amazonaws.com/aws-appmesh-envoy:v1.34.13.0-prod",
  "essential": true,
  "environment": [{
    "name": "APPMESH_RESOURCE_ARN",
    "value": "arn:aws:appmesh:us-west-2:111122223333:mesh/apps/virtualNode/serviceB"
  }],
  "healthCheck": {
    "command": [
      "CMD-SHELL",
      "curl -s http://localhost:9901/server_info | grep state | grep -q LIVE"
    ],
    "startPeriod": 10,
    "interval": 5,
    "timeout": 2,
    "retries": 3
  },
  "user": "1337"
}
```

Contoh ketentuan tugas

Contoh berikut definisi tugas Amazon ECS menunjukkan cara menggabungkan contoh dari atas ke definisi tugas untuk `taskB`. Contoh disediakan untuk membuat tugas untuk kedua jenis peluncuran Amazon ECS dengan atau tanpa menggunakan AWS X-Ray. Ubah *replaceable* nilai, yang sesuai, untuk membuat definisi tugas untuk tugas bernama `taskBv2` dan `taskA` dari skenario. Gantikan nama mesh dan nama node virtual Anda dengan APPMESH_RESOURCE_ARN nilai dan daftar port yang didengarkan aplikasi Anda untuk `AppPorts` nilai konfigurasi proxy. Secara default, App Mesh menggunakan nama sumber daya yang Anda tentukan APPMESH_RESOURCE_ARN saat Envoy merujuk dirinya sendiri dalam metrik dan jejak. Anda dapat menimpa perilaku ini dengan mengatur variabel lingkungan APPMESH_RESOURCE_CLUSTER dengan nama Anda sendiri. Semua

properti dalam contoh berikut diperlukan. Beberapa nilai properti juga diperlukan, tetapi beberapa di antaranya *replaceable*.

Jika Anda menjalankan tugas Amazon ECS seperti yang dijelaskan di bagian Kredensial, Anda perlu menambahkan [peran IAM tugas](#) yang ada, ke contoh.

Important

Fargate harus menggunakan nilai port yang lebih besar dari 1024.

Example Definisi tugas JSON untuk Amazon ECS - Fargate pada wadah Linux

```
{
  "family" : "taskB",
  "memory" : "1024",
  "cpu" : "0.5 vCPU",
  "proxyConfiguration" : {
    "containerName" : "envoy",
    "properties" : [
      {
        "name" : "ProxyIngressPort",
        "value" : "15000"
      },
      {
        "name" : "AppPorts",
        "value" : "9080"
      },
      {
        "name" : "EgressIgnoredIPs",
        "value" : "169.254.170.2,169.254.169.254"
      },
      {
        "name": "EgressIgnoredPorts",
        "value": "22"
      },
      {
        "name" : "IgnoredUID",
        "value" : "1337"
      },
      {
        "name" : "ProxyEgressPort",
```

```
        "value" : "15001"
      }
    ],
    "type" : "APPMESH"
  },
  "containerDefinitions" : [
    {
      "name" : "appName",
      "image" : "appImage",
      "portMappings" : [
        {
          "containerPort" : 9080,
          "protocol" : "tcp"
        }
      ],
      "essential" : true,
      "dependsOn" : [
        {
          "containerName" : "envoy",
          "condition" : "HEALTHY"
        }
      ]
    },
    {
      "name" : "envoy",
      "image" : "840364872350.dkr.ecr.us-west-2.amazonaws.com/aws-appmesh-
envoy:v1.34.13.0-prod",
      "essential" : true,
      "environment" : [
        {
          "name" : "APPMESH_VIRTUAL_NODE_NAME",
          "value" : "mesh/apps/virtualNode/serviceB"
        }
      ]
    },
    {
      "healthCheck" : {
        "command" : [
          "CMD-SHELL",
          "curl -s http://localhost:9901/server_info | grep state | grep -q LIVE"
        ],
        "interval" : 5,
        "retries" : 3,
        "startPeriod" : 10,
        "timeout" : 2
      }
    }
  ],
```

```

        "memory" : 500,
        "user" : "1337"
    }
],
"requiresCompatibilities" : [ "FARGATE" ],
"taskRoleArn" : "arn:aws:iam::<123456789012>:role/ecsTaskRole",
"executionRoleArn" : "arn:aws:iam::<123456789012>:role/ecsTaskExecutionRole",
"networkMode" : "awsvpc"
}

```

Example Definisi tugas JSON untuk Amazon ECS dengan - AWS X-Ray Fargate pada wadah Linux

X-Ray memungkinkan Anda mengumpulkan data tentang permintaan yang dilayani aplikasi dan menyediakan alat yang dapat Anda gunakan untuk memvisualisasikan arus lalu lintas. Menggunakan driver X-Ray untuk Utusan memungkinkan Utusan untuk melaporkan informasi penelusuran ke X-Ray. Anda dapat mengaktifkan penelusuran X-Ray menggunakan konfigurasi [Envoy](#). Berdasarkan konfigurasi, Envoy mengirimkan data penelusuran ke daemon X-Ray yang berjalan sebagai wadah [sespan](#) dan daemon meneruskan jejak ke layanan X-Ray. Setelah jejak dipublikasikan ke X-Ray, Anda dapat menggunakan konsol X-Ray untuk memvisualisasikan grafik panggilan layanan dan meminta detail jejak. JSON berikut merupakan definisi tugas untuk mengaktifkan integrasi X-Ray.

```

{

"family" : "taskB",
"memory" : "1024",
"cpu" : "512",
"proxyConfiguration" : {
    "containerName" : "envoy",
    "properties" : [
        {
            "name" : "ProxyIngressPort",
            "value" : "15000"
        },
        {
            "name" : "AppPorts",
            "value" : "9080"
        },
        {
            "name" : "EgressIgnoredIPs",
            "value" : "169.254.170.2,169.254.169.254"
        }
    ]
}
}

```

```

    {
      "name": "EgressIgnoredPorts",
      "value": "22"
    },
    {
      "name" : "IgnoredUID",
      "value" : "1337"
    },
    {
      "name" : "ProxyEgressPort",
      "value" : "15001"
    }
  ],
  "type" : "APPMESH"
},
"containerDefinitions" : [
  {
    "name" : "appName",
    "image" : "appImage",
    "portMappings" : [
      {
        "containerPort" : 9080,
        "protocol" : "tcp"
      }
    ],
    "essential" : true,
    "dependsOn" : [
      {
        "containerName" : "envoy",
        "condition" : "HEALTHY"
      }
    ]
  },
  {
    "name" : "envoy",
    "image" : "840364872350.dkr.ecr.us-west-2.amazonaws.com/aws-appmesh-
envoy:v1.34.13.0-prod",
    "essential" : true,
    "environment" : [
      {
        "name" : "APPMESH_VIRTUAL_NODE_NAME",
        "value" : "mesh/apps/virtualNode/serviceB"
      }
    ],
  },

```

```

    {
      "name": "ENABLE_ENVOY_XRAY_TRACING",
      "value": "1"
    }
  ],
  "healthCheck" : {
    "command" : [
      "CMD-SHELL",
      "curl -s http://localhost:9901/server_info | grep state | grep -q LIVE"
    ],
    "interval" : 5,
    "retries" : 3,
    "startPeriod" : 10,
    "timeout" : 2
  },
  "memory" : 500,
  "user" : "1337"
},
{
  "name" : "xray-daemon",
  "image" : "amazon/aws-xray-daemon",
  "user" : "1337",
  "essential" : true,
  "cpu" : "32",
  "memoryReservation" : "256",
  "portMappings" : [
    {
      "containerPort" : 2000,
      "protocol" : "udp"
    }
  ]
}
],
"requiresCompatibilities" : [ "FARGATE" ],
"taskRoleArn" : "arn:aws:iam::<123456789012>:role/ecsTaskRole",
"executionRoleArn" : "arn:aws:iam::<123456789012>:role/ecsTaskExecutionRole",
"networkMode" : "awsvpc"
}

```

Example Definisi tugas JSON untuk Amazon ECS - Jenis peluncuran EC2

```

{
  "family": "taskB",

```

```
"memory": "256",
"proxyConfiguration": {
  "type": "APPMESH",
  "containerName": "envoy",
  "properties": [
    {
      "name": "IgnoredUID",
      "value": "1337"
    },
    {
      "name": "ProxyIngressPort",
      "value": "15000"
    },
    {
      "name": "ProxyEgressPort",
      "value": "15001"
    },
    {
      "name": "AppPorts",
      "value": "9080"
    },
    {
      "name": "EgressIgnoredIPs",
      "value": "169.254.170.2,169.254.169.254"
    },
    {
      "name": "EgressIgnoredPorts",
      "value": "22"
    }
  ]
},
"containerDefinitions": [
  {
    "name": "appName",
    "image": "appImage",
    "portMappings": [
      {
        "containerPort": 9080,
        "hostPort": 9080,
        "protocol": "tcp"
      }
    ],
    "essential": true,
    "dependsOn": [
```

```

    {
      "containerName": "envoy",
      "condition": "HEALTHY"
    }
  ]
},
{
  "name": "envoy",
  "image": "840364872350.dkr.ecr.us-west-2.amazonaws.com/aws-appmesh-
envoy:v1.34.13.0-prod",
  "essential": true,
  "environment": [
    {
      "name": "APPMESH_VIRTUAL_NODE_NAME",
      "value": "mesh/apps/virtualNode/serviceB"
    }
  ],
  "healthCheck": {
    "command": [
      "CMD-SHELL",
      "curl -s http://localhost:9901/server_info | grep state | grep -q LIVE"
    ],
    "startPeriod": 10,
    "interval": 5,
    "timeout": 2,
    "retries": 3
  },
  "user": "1337"
}
],
"requiresCompatibilities" : [ "EC2" ],
"taskRoleArn" : "arn:aws:iam::123456789012:role/ecsTaskRole",
"executionRoleArn" : "arn:aws:iam::123456789012:role/ecsTaskExecutionRole",
"networkMode": "awsvpc"
}

```

Example Definisi tugas JSON untuk Amazon ECS dengan AWS X-Ray - tipe peluncuran EC2

```

{
  "family": "taskB",
  "memory": "256",
  "cpu" : "1024",
  "proxyConfiguration": {

```

```
"type": "APPMESH",
"containerName": "envoy",
"properties": [
  {
    "name": "IgnoredUID",
    "value": "1337"
  },
  {
    "name": "ProxyIngressPort",
    "value": "15000"
  },
  {
    "name": "ProxyEgressPort",
    "value": "15001"
  },
  {
    "name": "AppPorts",
    "value": "9080"
  },
  {
    "name": "EgressIgnoredIPs",
    "value": "169.254.170.2,169.254.169.254"
  },
  {
    "name": "EgressIgnoredPorts",
    "value": "22"
  }
]
},
"containerDefinitions": [
  {
    "name": "appName",
    "image": "appImage",
    "portMappings": [
      {
        "containerPort": 9080,
        "hostPort": 9080,
        "protocol": "tcp"
      }
    ],
    "essential": true,
    "dependsOn": [
      {
        "containerName": "envoy",
```

```
        "condition": "HEALTHY"
      }
    ]
  },
  {
    "name": "envoy",
    "image": "840364872350.dkr.ecr.us-west-2.amazonaws.com/aws-appmesh-
envoy:v1.34.13.0-prod",
    "essential": true,
    "environment": [
      {
        "name": "APPMESH_VIRTUAL_NODE_NAME",
        "value": "mesh/apps/virtualNode/serviceB"
      },
      {
        "name": "ENABLE_ENVOY_XRAY_TRACING",
        "value": "1"
      }
    ],
    "healthCheck": {
      "command": [
        "CMD-SHELL",
        "curl -s http://localhost:9901/server_info | grep state | grep -q LIVE"
      ],
      "startPeriod": 10,
      "interval": 5,
      "timeout": 2,
      "retries": 3
    },
    "user": "1337"
  },
  {
    "name": "xray-daemon",
    "image": "amazon/aws-xray-daemon",
    "user": "1337",
    "essential": true,
    "cpu": 32,
    "memoryReservation": 256,
    "portMappings": [
      {
        "containerPort": 2000,
        "protocol": "udp"
      }
    ]
  }
]
```

```
    }  
  ],  
  "requiresCompatibilities" : [ "EC2" ],  
  "taskRoleArn" : "arn:aws:iam::123456789012:role/ecsTaskRole",  
  "executionRoleArn" : "arn:aws:iam::123456789012:role/ecsTaskExecutionRole",  
  "networkMode": "awsvpc"  
}
```

Topik lanjutan

Penerapan Canary menggunakan App Mesh

Penerapan dan rilis Canary membantu Anda mengalihkan lalu lintas antara versi lama aplikasi dan versi yang baru digunakan. Ini juga memantau kesehatan versi yang baru digunakan. Jika ada masalah dengan versi baru, penyebaran kenari dapat secara otomatis mengalihkan lalu lintas kembali ke versi lama. Penerapan Canary memberi Anda kemampuan untuk mengalihkan lalu lintas antar versi aplikasi dengan kontrol lebih besar.

Untuk informasi selengkapnya tentang cara menerapkan penerapan canary untuk Amazon ECS menggunakan App Mesh, lihat [Membuat pipeline dengan penerapan canary untuk Amazon ECS menggunakan App Mesh](#)

Note

Untuk contoh dan penelusuran lainnya untuk App Mesh, lihat repositori contoh [App Mesh](#).

Memulai dengan AWS App Mesh dan Kubernetes

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk. AWS App Mesh Setelah 30 September 2026, Anda tidak akan lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Ketika Anda berintegrasi AWS App Mesh dengan Kubernetes menggunakan pengontrol App Mesh untuk Kubernetes, Anda mengelola resource App Mesh, seperti mesh, layanan virtual, node virtual, router virtual, dan rute melalui Kubernetes. Anda juga secara otomatis menambahkan gambar container sidecar App Mesh ke spesifikasi pod Kubernetes. Tutorial ini memandu Anda melalui instalasi pengontrol App Mesh untuk Kubernetes untuk mengaktifkan integrasi ini.

Pengontrol disertai dengan penerapan definisi sumber daya kustom Kubernetes berikut: `meshes`, `virtual services` dan `virtual nodes` `virtual routers`. Pengontrol mengawasi pembuatan, modifikasi, dan penghapusan sumber daya kustom dan membuat perubahan pada sumber daya App Mesh [the section called “Jerat”](#), [the section called “Layanan virtual”](#), [the section called “Node virtual”](#), [the section called “Gerbang virtual”](#), [the section called “Rute gerbang”](#), [the section called “Router virtual”](#) (termasuk [the section called “Rute”](#)) yang sesuai melalui App Mesh API. Untuk mempelajari lebih lanjut atau berkontribusi pada pengontrol, lihat [GitHub proyek](#).

Controller juga menginstal webhook yang menyuntikkan kontainer berikut ke dalam pod Kubernetes yang diberi label dengan nama yang Anda tentukan.

- Proxy App Mesh Envoy — Envoy menggunakan konfigurasi yang ditentukan dalam bidang kontrol App Mesh untuk menentukan ke mana harus mengirim lalu lintas aplikasi Anda.
- Manajer rute proxy App Mesh — Memperbarui `iptables` aturan di namespace jaringan pod yang merutekan lalu lintas masuk dan keluar melalui Envoy. Container ini berjalan sebagai kontainer init Kubernetes di dalam pod.

Prasyarat

- Pemahaman yang ada tentang konsep App Mesh. Untuk informasi selengkapnya, lihat [Apa itu AWS App Mesh?](#).
- Pemahaman yang ada tentang konsep Kubernetes. Untuk informasi selengkapnya, lihat [Apa itu Kubernetes dalam dokumentasi Kubernetes](#).
- Cluster Kubernetes yang sudah ada. Jika Anda tidak memiliki klaster yang ada, lihat [Memulai Amazon EKS](#) di Panduan Pengguna Amazon EKS. Jika Anda menjalankan cluster Kubernetes Anda sendiri di Amazon EC2, pastikan Docker diautentikasi ke repositori Amazon ECR tempat image Envoy berada. Untuk informasi selengkapnya, lihat [Envoy image](#), [Registry authentication](#) di Amazon Elastic Container Registry User Guide, dan [Pull an Image dari Private Registry](#) di dokumentasi Kubernetes.
- App Mesh mendukung layanan Linux yang terdaftar dengan DNS, AWS Cloud Map, atau keduanya. Untuk menggunakan panduan memulai ini, kami sarankan Anda memiliki tiga layanan

yang sudah ada yang terdaftar di DNS. Prosedur dalam topik ini mengasumsikan bahwa layanan yang ada diberi nama `serviceA`, `serviceB`, `serviceBv2` dan bahwa semua layanan dapat ditemukan melalui namespace bernama `apps.local`

Anda dapat membuat mesh layanan dan sumber dayanya bahkan jika layanan tidak ada, tetapi Anda tidak dapat menggunakan mesh sampai Anda telah menerapkan layanan yang sebenarnya.

- AWS CLI Versi 1.18.116 atau yang lebih baru atau 2.0.38 atau yang lebih baru diinstal. Untuk menginstal atau memutakhirkan AWS CLI, lihat [Menginstal AWS CLI](#).
- `kubectl` klien yang dikonfigurasi untuk berkomunikasi dengan kluster Kubernetes Anda. Jika Anda menggunakan Amazon Elastic Kubernetes Service, Anda dapat menggunakan petunjuk untuk [kubectl](#) menginstal dan mengonfigurasi file `kubeconfig`
- Helm versi 3.0 atau yang lebih baru diinstal. Jika Anda belum menginstal Helm, lihat [Menggunakan Helm dengan Amazon EKS](#) di Panduan Pengguna Amazon EKS.
- Amazon EKS saat ini hanya mendukung `IPv4_ONLY` dan `IPv6_ONLY` hanya preferensi IP, karena Amazon EKS saat ini hanya mendukung pod yang mampu melayani hanya IPv4 lalu lintas atau hanya IPv6 lalu lintas.

Langkah-langkah yang tersisa mengasumsikan bahwa layanan yang sebenarnya diberi nama `serviceA`, `serviceB`, `serviceBv2` dan bahwa semua layanan dapat ditemukan melalui namespace bernama `apps.local`

Langkah 1: Instal komponen integrasi

Instal komponen integrasi satu kali ke setiap cluster yang meng-host pod yang ingin Anda gunakan dengan App Mesh.

Untuk menginstal komponen integrasi

1. Langkah-langkah yang tersisa dari prosedur ini memerlukan cluster tanpa versi pra-rilis dari controller diinstal. Jika Anda telah menginstal versi pra-rilis, atau tidak yakin apakah sudah, Anda dapat mengunduh dan menjalankan skrip yang memeriksa untuk melihat apakah versi pra-rilis diinstal pada kluster Anda.

```
curl -o pre_upgrade_check.sh https://raw.githubusercontent.com/aws/eks-charts/master/stable/appmesh-controller/upgrade/pre_upgrade_check.sh
sh ./pre_upgrade_check.sh
```

Jika skrip kembali `Your cluster is ready for upgrade`. Please proceed to the installation instructions maka Anda dapat melanjutkan ke langkah berikutnya. Jika pesan yang berbeda dikembalikan, maka Anda harus menyelesaikan langkah-langkah peningkatan sebelum melanjutkan. Untuk informasi selengkapnya tentang memutakhirkan versi pra-rilis, lihat [Upgrade](#) pada GitHub

2. Tambahkan eks-charts repositori ke Helm.

```
helm repo add eks https://aws.github.io/eks-charts
```

3. Instal definisi sumber daya kustom (CRD) App Mesh Kubernetes.

```
kubectl apply -k "https://github.com/aws/eks-charts/stable/appmesh-controller/crds?ref=master"
```

4. Buat namespace Kubernetes untuk controller.

```
kubectl create ns appmesh-system
```

5. Tetapkan variabel berikut untuk digunakan dalam langkah selanjutnya. Ganti *cluster-name* dan *Region-code* dengan nilai untuk cluster Anda yang ada.

```
export CLUSTER_NAME=cluster-name
export AWS_REGION=Region-code
```

6. (Opsional) Jika Anda ingin menjalankan controller di Fargate, maka Anda perlu membuat profil Fargate. Jika Anda belum `eksctl` menginstal, lihat [Menginstal atau Memutakhirkan eksctl](#) di Panduan Pengguna Amazon EKS. Jika Anda lebih suka membuat profil menggunakan konsol, lihat [Membuat profil Fargate](#) di Panduan Pengguna Amazon EKS.

```
eksctl create fargateprofile --cluster $CLUSTER_NAME --name appmesh-system --namespace appmesh-system
```

7. Buat penyedia identitas OpenID Connect (OIDC) untuk klaster Anda. Jika Anda belum `eksctl` menginstal, Anda dapat menginstalnya dengan petunjuk di [Menginstal atau memutakhirkan eksctl](#) di Panduan Pengguna Amazon EKS. Jika Anda lebih suka membuat penyedia menggunakan konsol, lihat [Mengaktifkan peran IAM untuk akun layanan di klaster Anda di Panduan Pengguna Amazon EKS](#).

```
eksctl utils associate-iam-oidc-provider \
```

```
--region=$AWS_REGION \  
--cluster $CLUSTER_NAME \  
--approve
```

8. Buat peran IAM, lampirkan [AWSAppMeshFullAccess](#) dan [AWSCloudMapFullAccess](#) AWS kelola kebijakan ke dalamnya, dan ikat ke akun layanan appmesh-controller Kubernetes. Peran ini memungkinkan pengontrol untuk menambah, menghapus, dan mengubah resource App Mesh.

Note

Perintah membuat peran AWS IAM dengan nama yang dibuat secara otomatis. Anda tidak dapat menentukan nama peran IAM yang dibuat.


```
eksctl create iamserviceaccount \  
--cluster $CLUSTER_NAME \  
--namespace appmesh-system \  
--name appmesh-controller \  
--attach-policy-arn arn:aws:iam::aws:policy/  
AWSCloudMapFullAccess,arn:aws:iam::aws:policy/AWSAppMeshFullAccess \  
--override-existing-serviceaccounts \  
--approve
```

Jika Anda lebih suka membuat akun layanan menggunakan Konsol Manajemen AWS atau AWS CLI, lihat [Membuat peran dan kebijakan IAM untuk akun layanan Anda](#) di Panduan Pengguna Amazon EKS. Jika Anda menggunakan Konsol Manajemen AWS atau AWS CLI untuk membuat akun, Anda juga perlu memetakan peran tersebut ke akun layanan Kubernetes. Untuk informasi selengkapnya, lihat [Menentukan peran IAM untuk akun layanan Anda](#) di Panduan Pengguna Amazon EKS.

9. Menerapkan pengontrol App Mesh. Untuk daftar semua opsi konfigurasi, lihat [Konfigurasi](#) aktif GitHub.
 1. Untuk menerapkan pengontrol App Mesh untuk klaster pribadi, Anda harus mengaktifkan App Mesh dan penemuan layanan titik akhir Amazon VPC Amazon ke subnet pribadi yang ditautkan terlebih dahulu. Anda juga diminta untuk mengatur `accountId`.

```
--set accountId=$AWS_ACCOUNT_ID
```

Untuk mengaktifkan penelusuran X-Ray di cluster pribadi, aktifkan titik akhir X-Ray dan Amazon ECR Amazon VPC. Pengontrol menggunakan secara `public.ecr.aws/xray/aws-xray-daemon:latest default`, jadi tarik gambar ini ke lokal dan [dorong ke repositori ECR pribadi Anda](#).

 Note

[Titik akhir Amazon VPC](#) saat ini tidak mendukung repositori publik Amazon ECR.

Contoh berikut menunjukkan penerapan controller dengan konfigurasi untuk X-Ray.

```
helm upgrade -i appmesh-controller eks/appmesh-controller \
  --namespace appmesh-system \
  --set region=$AWS_REGION \
  --set serviceAccount.create=false \
  --set serviceAccount.name=appmesh-controller \
  --set accountId=$AWS_ACCOUNT_ID \
  --set log.level=debug \
  --set tracing.enabled=true \
  --set tracing.provider=x-ray \
  --set xray.image.repository=your-account-id.dkr.ecr.your-region.amazonaws.com/your-repository \
  --set xray.image.tag=your-xray-daemon-image-tag
```

Verifikasi apakah daemon X-Ray berhasil disuntikkan saat mengikat penerapan aplikasi dengan node atau gateway virtual Anda.

Untuk informasi selengkapnya, lihat [Cluster Pribadi](#) di Panduan Pengguna Amazon EKS.

2. Menerapkan pengontrol App Mesh untuk cluster lain. Untuk daftar semua opsi konfigurasi, lihat [Konfigurasi](#) aktif GitHub.

```
helm upgrade -i appmesh-controller eks/appmesh-controller \
  --namespace appmesh-system \
  --set region=$AWS_REGION \
  --set serviceAccount.create=false \
  --set serviceAccount.name=appmesh-controller
```

Note

Jika keluarga cluster Amazon EKS Anda IPv6, setel nama cluster saat menerapkan pengontrol App Mesh dengan menambahkan opsi berikut ke perintah `--set clusterName=$CLUSTER_NAME` sebelumnya.

Important

Jika cluster Anda berada di `me-south-1`, `ap-east-1`, `ap-southeast-3`, `eu-south-1`, `il-central-1`, atau `af-south-1` Regions, maka Anda perlu menambahkan opsi berikut ke perintah sebelumnya:

Ganti *account-id* dan *Region-code* dengan salah satu set nilai yang sesuai.

- Untuk gambar sespan:

- ```
--set image.repository=account-id.dkr.ecr.Region-code.amazonaws.com/
amazon/appmesh-controller
```
- `772975370895.dkr.ecr.me-south-1.amazonaws.com /:v1.34.13.0-prod aws-appmesh-envoy`
- `856666278305.dkr.ecr.ap-east-1.amazonaws.com /:v1.34.13.0-prod aws-appmesh-envoy`
- `909464085924.dkr.ecr.ap-southeast-3.amazonaws.com /:v1.34.13.0-prod aws-appmesh-envoy`
- `422531588944.dkr.ecr.eu-south-1.amazonaws.com /:v1.34.13.0-prod aws-appmesh-envoy`
- `564877687649.dkr.ecr.il-central-1.amazonaws.com /:v1.34.13.0-prod aws-appmesh-envoy`
- `924023996002.dkr.ecr.af-south-1.amazonaws.com /:v1.34.13.0-prod aws-appmesh-envoy`
- Gambar yang lebih tua URIs dapat ditemukan di [log perubahan](#) GitHub. AWS Akun tempat gambar hadir telah berubah dalam versi `1.5.0`. Versi gambar yang lebih lama di-host di AWS akun yang ditemukan di Amazon Elastic Kubernetes [Service](#) Amazon Container Image Registries.

- Untuk gambar pengontrol:
  - ```
--set sidecar.image.repository=account-id.dkr.ecr.Region-code.amazonaws.com/aws-appmesh-envoy
```
 - 772975370895.dkr.ecr.me-selatan-1.amazonaws.com/amazon/appmesh-pengontrol: v1.13.1
 - 856666278305.dkr.ecr.ap-east-1.amazonaws.com/amazon/appmesh-pengontrol: v1.13.1
 - 909464085924.dkr.ecr.ap-southeast-3.amazonaws.com/amazon/appmesh-pengontrol: v1.13.1
 - 422531588944.dkr.ecr.eu-south-1.amazonaws.com/amazon/appmesh-pengontrol: v1.13.1
 - 564877687649.dkr.ecr.il-central-1.amazonaws.com/amazon/appmesh-pengontrol: v1.13.1
 - 924023996002.dkr.ecr.af-selatan-1.amazonaws.com/amazon/appmesh-pengontrol: v1.13.1
- Untuk gambar init sidecar:
 - ```
--set sidecar.image.repository=account-id.dkr.ecr.Region-code.amazonaws.com/aws-appmesh-envoy
```
  - 772975370895.dkr.ecr.me-south-1.amazonaws.com/-manajer:v7-prod aws-appmesh-proxy-route
  - 856666278305.dkr.ecr.ap-east-1.amazonaws.com/-manajer:v7-prod aws-appmesh-proxy-route
  - 909464085924.dkr.ecr.ap-southeast-3.amazonaws.com/-manajer:v7-prod aws-appmesh-proxy-route
  - 422531588944.dkr.ecr.eu-south-1.amazonaws.com/-manajer:v7-prod aws-appmesh-proxy-route
  - 564877687649.dkr.ecr.il-central-1.amazonaws.com/-manajer:v7-prod aws-appmesh-proxy-route
  - 924023996002.dkr.ecr.af-south-1.amazonaws.com/-manajer:v7-prod aws-appmesh-proxy-route

**⚠ Important**

Hanya versi v1.9.0.0-prod atau yang lebih baru yang didukung untuk digunakan dengan App Mesh.

10. Konfirmasikan bahwa versi pengontrol adalah v1.4.0 atau lebih baru. Anda dapat meninjau [log perubahan](#) GitHub.

```
kubectl get deployment appmesh-controller \
 -n appmesh-system \
 -o json | jq -r ".spec.template.spec.containers[].image" | cut -f2 -d ':'
```

**ℹ Note**

Jika Anda melihat log untuk wadah yang sedang berjalan, Anda mungkin melihat baris yang menyertakan teks berikut, yang dapat diabaikan dengan aman.

```
Neither -kubeconfig nor -master was specified. Using the inClusterConfig.
This might not work.
```

## Langkah 2: Menyebarkan sumber daya App Mesh

Saat Anda menerapkan aplikasi di Kubernetes, Anda juga membuat sumber daya kustom Kubernetes sehingga pengontrol dapat membuat resource App Mesh yang sesuai. Prosedur berikut membantu Anda menerapkan resource App Mesh dengan beberapa fiturnya. Anda dapat menemukan contoh manifes untuk menerapkan fitur resource App Mesh lainnya di v1beta2 sub-folder dari banyak folder fitur yang tercantum pada [penelusuran App Mesh](#). GitHub

**⚠ Important**

Setelah pengontrol membuat resource App Mesh, sebaiknya Anda hanya membuat perubahan atau menghapus resource App Mesh menggunakan controller. Jika Anda membuat perubahan atau menghapus sumber daya menggunakan App Mesh, pengontrol tidak akan mengubah atau membuat ulang sumber daya App Mesh yang diubah atau

dihapus selama sepuluh jam, secara default. Anda dapat mengonfigurasi durasi ini menjadi lebih sedikit. Untuk informasi selengkapnya, lihat [Konfigurasi](#) pada GitHub.

Untuk menerapkan sumber daya App Mesh

1. Buat namespace Kubernetes untuk menerapkan resource App Mesh.
  - a. Simpan konten berikut ini ke file bernama `namespace.yaml` pada komputer Anda.

```
apiVersion: v1
kind: Namespace
metadata:
 name: my-apps
 labels:
 mesh: my-mesh
 appmesh.k8s.aws/sidecarInjectorWebhook: enabled
```

- b. Buat namespace.

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

2. Buat mesh layanan App Mesh.
  - a. Simpan konten berikut ini ke file bernama `mesh.yaml` pada komputer Anda. File ini digunakan untuk membuat sumber daya mesh bernama *my-mesh*. Mesh layanan adalah batas logis untuk lalu lintas jaringan antara layanan yang berada di dalamnya.

```
apiVersion: appmesh.k8s.aws/v1beta2
kind: Mesh
metadata:
 name: my-mesh
spec:
 namespaceSelector:
 matchLabels:
 mesh: my-mesh
```

- b. Buat mesh.

```
kubectl apply -f mesh.yaml
```

- c. Lihat detail sumber daya mesh Kubernetes yang telah dibuat.

```
kubectl describe mesh my-mesh
```

## Output

```
Name: my-mesh
Namespace:
Labels: <none>
Annotations: kubectl.kubernetes.io/last-applied-configuration:
 {"apiVersion":"appmesh.k8s.aws/v1beta2","kind":"Mesh","metadata":{"annotations":{},"name":"my-mesh"},"spec":
 {"namespaceSelector":{"matchLa...
API Version: appmesh.k8s.aws/v1beta2
Kind: Mesh
Metadata:
 Creation Timestamp: 2020-06-17T14:51:37Z
 Finalizers:
 finalizers.appmesh.k8s.aws/mesh-members
 finalizers.appmesh.k8s.aws/aws-appmesh-resources
 Generation: 1
 Resource Version: 6295
 Self Link: /apis/appmesh.k8s.aws/v1beta2/meshes/my-mesh
 UID: 111a11b1-c11d-1e1f-gh1i-j11k11111m711
Spec:
 Aws Name: my-mesh
 Namespace Selector:
 Match Labels:
 Mesh: my-mesh
Status:
 Conditions:
 Last Transition Time: 2020-06-17T14:51:37Z
 Status: True
 Type: MeshActive
 Mesh ARN: arn:aws:appmesh:us-west-2:111122223333:mesh/my-mesh
 Observed Generation: 1
Events: <none>
```

- d. Lihat detail tentang mesh layanan App Mesh yang dibuat oleh pengontrol.

```
aws appmesh describe-mesh --mesh-name my-mesh
```

## Output

```
{
 "mesh": {
 "meshName": "my-mesh",
 "metadata": {
 "arn": "arn:aws:appmesh:us-west-2:111122223333:mesh/my-mesh",
 "createdAt": "2020-06-17T09:51:37.920000-05:00",
 "lastUpdatedAt": "2020-06-17T09:51:37.920000-05:00",
 "meshOwner": "111122223333",
 "resourceOwner": "111122223333",
 "uid": "111a11b1-c11d-1e1f-gh1i-j11k11111m711",
 "version": 1
 },
 "spec": {},
 "status": {
 "status": "ACTIVE"
 }
 }
}
```

3. Buat node virtual App Mesh. Node virtual bertindak sebagai pointer logis ke penerapan Kubernetes.
  - a. Simpan konten berikut ini ke file bernama `virtual-node.yaml` pada komputer Anda. File ini digunakan untuk membuat node virtual App Mesh bernama `my-service-a` di `my-apps` namespace. Node virtual mewakili layanan Kubernetes yang dibuat di langkah selanjutnya. Nilai untuk `hostname` adalah nama host DNS yang sepenuhnya memenuhi syarat dari layanan aktual yang diwakili oleh node virtual ini.

```
apiVersion: appmesh.k8s.aws/v1beta2
kind: VirtualNode
metadata:
 name: my-service-a
 namespace: my-apps
spec:
 podSelector:
 matchLabels:
 app: my-app-1
 listeners:
 - portMapping:
 port: 80
 protocol: http
```

```
serviceDiscovery:
 dns:
 hostname: my-service-a.my-apps.svc.cluster.local
```

Node virtual memiliki kemampuan, seperti end-to-end enkripsi dan pemeriksaan kesehatan, yang tidak tercakup dalam tutorial ini. Untuk informasi selengkapnya, lihat [the section called “Node virtual”](#). Untuk melihat semua pengaturan yang tersedia untuk node virtual yang dapat Anda atur dalam spesifikasi sebelumnya, jalankan perintah berikut.

```
aws appmesh create-virtual-node --generate-cli-skeleton yml-input
```

- b. Menyebarkan simpul virtual.

```
kubectl apply -f virtual-node.yml
```

- c. Lihat detail sumber daya node virtual Kubernetes yang telah dibuat.

```
kubectl describe virtualnode my-service-a -n my-apps
```

## Output

```
Name: my-service-a
Namespace: my-apps
Labels: <none>
Annotations: kubectl.kubernetes.io/last-applied-configuration:
 {"apiVersion":"appmesh.k8s.aws/v1beta2", "kind":"VirtualNode", "metadata":{"annotations":{}}, "name":"my-service-a", "namespace":"my-apps"}, "s...
API Version: appmesh.k8s.aws/v1beta2
Kind: VirtualNode
Metadata:
 Creation Timestamp: 2020-06-17T14:57:29Z
 Finalizers:
 finalizers.appmesh.k8s.aws/aws-appmesh-resources
 Generation: 2
 Resource Version: 22545
 Self Link: /apis/appmesh.k8s.aws/v1beta2/namespaces/my-apps/virtualnodes/my-service-a
 UID: 111a11b1-c11d-1e1f-gh1i-j11k11111m711
Spec:
 Aws Name: my-service-a_my-apps
```

```
Listeners:
 Port Mapping:
 Port: 80
 Protocol: http
 Mesh Ref:
 Name: my-mesh
 UID: 111a11b1-c11d-1e1f-gh1i-j11k11111m711
 Pod Selector:
 Match Labels:
 App: nginx
 Service Discovery:
 Dns:
 Hostname: my-service-a.my-apps.svc.cluster.local
Status:
 Conditions:
 Last Transition Time: 2020-06-17T14:57:29Z
 Status: True
 Type: VirtualNodeActive
 Observed Generation: 2
 Virtual Node ARN: arn:aws:appmesh:us-west-2:111122223333:mesh/my-mesh/virtualNode/my-service-a_my-apps
 Events: <none>
```

- d. Lihat detail node virtual yang dibuat pengontrol di App Mesh.

#### Note

Meskipun nama node virtual yang dibuat di Kubernetes adalah *my-service-a*, nama node virtual yang dibuat di App Mesh adalah *my-service-a\_my-apps*. Kontroler menambahkan nama namespace Kubernetes ke nama node virtual App Mesh saat membuat resource App Mesh. Nama namespace ditambahkan karena di Kubernetes Anda dapat membuat node virtual dengan nama yang sama di ruang nama yang berbeda, tetapi di App Mesh nama node virtual harus unik di dalam mesh.

```
aws appmesh describe-virtual-node --mesh-name my-mesh --virtual-node-name my-service-a_my-apps
```

#### Output

```

{
 "virtualNode": {
 "meshName": "my-mesh",
 "metadata": {
 "arn": "arn:aws:appmesh:us-west-2:111122223333:mesh/my-mesh/virtualNode/my-service-a_my-apps",
 "createdAt": "2020-06-17T09:57:29.840000-05:00",
 "lastUpdatedAt": "2020-06-17T09:57:29.840000-05:00",
 "meshOwner": "111122223333",
 "resourceOwner": "111122223333",
 "uid": "111a11b1-c11d-1e1f-gh1i-j11k11111m711",
 "version": 1
 },
 "spec": {
 "backends": [],
 "listeners": [
 {
 "portMapping": {
 "port": 80,
 "protocol": "http"
 }
 }
],
 "serviceDiscovery": {
 "dns": {
 "hostname": "my-service-a.my-apps.svc.cluster.local"
 }
 }
 },
 "status": {
 "status": "ACTIVE"
 },
 "virtualNodeName": "my-service-a_my-apps"
 }
}

```

4. Buat router virtual App Mesh. Router virtual menangani lalu lintas untuk satu atau lebih layanan virtual dalam mesh Anda.
  - a. Simpan konten berikut ini ke file bernama `virtual-router.yaml` pada komputer Anda. File ini digunakan untuk membuat router virtual untuk merutekan lalu lintas ke node virtual bernama `my-service-a` yang dibuat pada langkah sebelumnya. Pengontrol

membuat router virtual App Mesh dan sumber daya rute. Anda dapat menentukan lebih banyak kemampuan untuk rute Anda dan menggunakan protokol selain `http`. Untuk informasi selengkapnya, lihat [the section called “Router virtual”](#) dan [the section called “Route”](#). Perhatikan bahwa nama node virtual yang direferensikan adalah nama node virtual Kubernetes, bukan nama node virtual App Mesh yang dibuat di App Mesh oleh controller.

```
apiVersion: appmesh.k8s.aws/v1beta2
kind: VirtualRouter
metadata:
 namespace: my-apps
 name: my-service-a-virtual-router
spec:
 listeners:
 - portMapping:
 port: 80
 protocol: http
 routes:
 - name: my-service-a-route
 httpRoute:
 match:
 prefix: /
 action:
 weightedTargets:
 - virtualNodeRef:
 name: my-service-a
 weight: 1
```

(Opsional) Untuk melihat semua pengaturan yang tersedia untuk router virtual yang dapat Anda atur dalam spesifikasi sebelumnya, jalankan perintah berikut.

```
aws appmesh create-virtual-router --generate-cli-skeleton yml-input
```

Untuk melihat semua pengaturan yang tersedia untuk rute yang dapat Anda atur dalam spesifikasi sebelumnya, jalankan perintah berikut.

```
aws appmesh create-route --generate-cli-skeleton yml-input
```

b. Menyebarkan router virtual.

```
kubectl apply -f virtual-router.yml
```

- c. Lihat sumber daya router virtual Kubernetes yang telah dibuat.

```
kubectl describe virtualrouter my-service-a-virtual-router -n my-apps
```

### Output dipersingkat

```
Name: my-service-a-virtual-router
Namespace: my-apps
Labels: <none>
Annotations: kubect1.kubernetes.io/last-applied-configuration:
 {"apiVersion":"appmesh.k8s.aws/v1beta2", "kind":"VirtualRouter", "metadata":{"annotations":{},"name":"my-
 service-a-virtual-router", "namespac...
API Version: appmesh.k8s.aws/v1beta2
Kind: VirtualRouter
...
Spec:
 Aws Name: my-service-a-virtual-router_my-apps
 Listeners:
 Port Mapping:
 Port: 80
 Protocol: http
 Mesh Ref:
 Name: my-mesh
 UID: 111a11b1-c11d-1e1f-gh1i-j11k11111m711
 Routes:
 Http Route:
 Action:
 Weighted Targets:
 Virtual Node Ref:
 Name: my-service-a
 Weight: 1
 Match:
 Prefix: /
 Name: my-service-a-route
Status:
 Conditions:
 Last Transition Time: 2020-06-17T15:14:01Z
 Status: True
 Type: VirtualRouterActive
 Observed Generation: 1
 Route AR Ns:
```

```

My - Service - A - Route: arn:aws:appmesh:us-west-2:111122223333:mesh/my-
mesh/virtualRouter/my-service-a-virtual-router_my-apps/route/my-service-a-route
Virtual Router ARN: arn:aws:appmesh:us-west-2:111122223333:mesh/my-
mesh/virtualRouter/my-service-a-virtual-router_my-apps
Events: <none>

```

- d. Lihat sumber daya router virtual yang dibuat oleh pengontrol di App Mesh. Anda menentukan `my-service-a-virtual-router_my-apps` untuk nama, karena ketika controller membuat router virtual di App Mesh, ia menambahkan nama namespace Kubernetes ke nama router virtual.

```

aws appmesh describe-virtual-router --virtual-router-name my-service-a-virtual-
router_my-apps --mesh-name my-mesh

```

## Output

```

{
 "virtualRouter": {
 "meshName": "my-mesh",
 "metadata": {
 "arn": "arn:aws:appmesh:us-west-2:111122223333:mesh/my-mesh/
virtualRouter/my-service-a-virtual-router_my-apps",
 "createdAt": "2020-06-17T10:14:01.547000-05:00",
 "lastUpdatedAt": "2020-06-17T10:14:01.547000-05:00",
 "meshOwner": "111122223333",
 "resourceOwner": "111122223333",
 "uid": "111a11b1-c11d-1e1f-gh1i-j11k11111m711",
 "version": 1
 },
 "spec": {
 "listeners": [
 {
 "portMapping": {
 "port": 80,
 "protocol": "http"
 }
 }
]
 },
 "status": {
 "status": "ACTIVE"
 }
 },

```

```

 "virtualRouterName": "my-service-a-virtual-router_my-apps"
 }
}

```

- e. Melihat sumber daya rute yang dibuat oleh pengontrol di App Mesh. Sumber daya rute tidak dibuat di Kubernetes karena rute tersebut merupakan bagian dari konfigurasi router virtual di Kubernetes. Informasi rute ditampilkan dalam detail sumber daya Kubernetes di sub-langkah. c Kontroler tidak menambahkan nama namespace Kubernetes ke nama rute App Mesh saat membuat rute di App Mesh karena nama rute unik untuk router virtual.

```

aws appmesh describe-route \
 --route-name my-service-a-route \
 --virtual-router-name my-service-a-virtual-router_my-apps \
 --mesh-name my-mesh

```

## Output

```

{
 "route": {
 "meshName": "my-mesh",
 "metadata": {
 "arn": "arn:aws:appmesh:us-west-2:111122223333:mesh/my-mesh/virtualRouter/my-service-a-virtual-router_my-apps/route/my-service-a-route",
 "createdAt": "2020-06-17T10:14:01.577000-05:00",
 "lastUpdatedAt": "2020-06-17T10:14:01.577000-05:00",
 "meshOwner": "111122223333",
 "resourceOwner": "111122223333",
 "uid": "111a11b1-c11d-1e1f-gh1i-j11k11111m711",
 "version": 1
 },
 "routeName": "my-service-a-route",
 "spec": {
 "httpRoute": {
 "action": {
 "weightedTargets": [
 {
 "virtualNode": "my-service-a_my-apps",
 "weight": 1
 }
]
 },
 "match": {

```

```

 "prefix": "/"
 }
 },
 "status": {
 "status": "ACTIVE"
 },
 "virtualRouterName": "my-service-a-virtual-router_my-apps"
 }
}

```

5. Buat layanan virtual App Mesh. Layanan virtual adalah abstraksi dari layanan sebenarnya yang disediakan oleh simpul virtual secara langsung atau tidak langsung berdasarkan alat router virtual. Layanan dependen memanggil layanan virtual Anda dengan namanya. Meskipun nama tidak masalah untuk App Mesh, kami sarankan untuk menamai layanan virtual nama domain yang sepenuhnya memenuhi syarat dari layanan aktual yang diwakili oleh layanan virtual. Dengan menamai layanan virtual Anda dengan cara ini, Anda tidak perlu mengubah kode aplikasi Anda untuk mereferensikan nama yang berbeda. Permintaan dirutekan ke node virtual atau router virtual yang ditentukan sebagai penyedia layanan virtual.
  - a. Simpan konten berikut ini ke file bernama `virtual-service.yaml` pada komputer Anda. File ini digunakan untuk membuat layanan virtual yang menggunakan penyedia router virtual untuk merutekan lalu lintas ke node virtual bernama `my-service-a` yang dibuat pada langkah sebelumnya. Nilai untuk `awsName` in spec adalah nama domain yang sepenuhnya memenuhi syarat (FQDN) dari layanan Kubernetes aktual yang diabstraksikan oleh layanan virtual ini. Layanan Kubernetes dibuat di [the section called “Langkah 3: Buat atau perbarui layanan”](#) Untuk informasi selengkapnya, lihat [the section called “Layanan virtual”](#).

```

apiVersion: appmesh.k8s.aws/v1beta2
kind: VirtualService
metadata:
 name: my-service-a
 namespace: my-apps
spec:
 awsName: my-service-a.my-apps.svc.cluster.local
 provider:
 virtualRouter:
 virtualRouterRef:
 name: my-service-a-virtual-router

```

Untuk melihat semua pengaturan yang tersedia untuk layanan virtual yang dapat Anda atur dalam spesifikasi sebelumnya, jalankan perintah berikut.

```
aws appmesh create-virtual-service --generate-cli-skeleton yml-input
```

- b. Buat layanan virtual.

```
kubectl apply -f virtual-service.yml
```

- c. Lihat detail sumber daya layanan virtual Kubernetes yang telah dibuat.

```
kubectl describe virtualservice my-service-a -n my-apps
```

## Output

```
Name: my-service-a
Namespace: my-apps
Labels: <none>
Annotations: kubectl.kubernetes.io/last-applied-configuration:
 {"apiVersion":"appmesh.k8s.aws/
v1beta2","kind":"VirtualService","metadata":{"annotations":{},"name":"my-
service-a","namespace":"my-apps"}}...
API Version: appmesh.k8s.aws/v1beta2
Kind: VirtualService
Metadata:
 Creation Timestamp: 2020-06-17T15:48:40Z
 Finalizers:
 finalizers.appmesh.k8s.aws/aws-appmesh-resources
 Generation: 1
 Resource Version: 13598
 Self Link: /apis/appmesh.k8s.aws/v1beta2/namespaces/my-apps/
virtualservices/my-service-a
 UID: 111a11b1-c11d-1e1f-gh1i-j11k11111m711
Spec:
 Aws Name: my-service-a.my-apps.svc.cluster.local
 Mesh Ref:
 Name: my-mesh
 UID: 111a11b1-c11d-1e1f-gh1i-j11k11111m711
 Provider:
 Virtual Router:
 Virtual Router Ref:
```

```

 Name: my-service-a-virtual-router
Status:
 Conditions:
 Last Transition Time: 2020-06-17T15:48:40Z
 Status: True
 Type: VirtualServiceActive
 Observed Generation: 1
 Virtual Service ARN: arn:aws:appmesh:us-west-2:111122223333:mesh/my-mesh/
virtualService/my-service-a.my-apps.svc.cluster.local
Events: <none>

```

- d. Lihat detail sumber daya layanan virtual yang dibuat oleh pengontrol di App Mesh. Kontroler Kubernetes tidak menambahkan nama namespace Kubernetes ke nama layanan virtual App Mesh ketika membuat layanan virtual di App Mesh karena nama layanan virtual adalah FQDN yang unik.

```

aws appmesh describe-virtual-service --virtual-service-name my-service-a.my-apps.svc.cluster.local --mesh-name my-mesh

```

## Output

```

{
 "virtualService": {
 "meshName": "my-mesh",
 "metadata": {
 "arn": "arn:aws:appmesh:us-west-2:111122223333:mesh/my-mesh/
virtualService/my-service-a.my-apps.svc.cluster.local",
 "createdAt": "2020-06-17T10:48:40.182000-05:00",
 "lastUpdatedAt": "2020-06-17T10:48:40.182000-05:00",
 "meshOwner": "111122223333",
 "resourceOwner": "111122223333",
 "uid": "111a11b1-c11d-1e1f-gh1i-j11k11111m711",
 "version": 1
 },
 "spec": {
 "provider": {
 "virtualRouter": {
 "virtualRouterName": "my-service-a-virtual-router_my-apps"
 }
 }
 },
 "status": {

```

```

 "status": "ACTIVE"
 },
 "virtualServiceName": "my-service-a.my-apps.svc.cluster.local"
 }
 }
}

```

Meskipun tidak tercakup dalam tutorial ini, controller juga dapat menyebarkan App Mesh [the section called “Gerbang virtual”](#) dan [the section called “Rute gerbang”](#). Untuk panduan penerapan sumber daya ini dengan pengontrol, lihat [Mengonfigurasi Gateway Masuk](#), atau [contoh manifes](#) yang menyertakan sumber daya pada. GitHub

### Langkah 3: Buat atau perbarui layanan

Pod apa pun yang ingin Anda gunakan dengan App Mesh harus memiliki wadah sidecar App Mesh yang ditambahkan ke dalamnya. Injektor secara otomatis menambahkan kontainer sidecar ke setiap pod yang digunakan dengan label yang Anda tentukan.

1. Aktifkan otorisasi proxy. Kami menyarankan Anda mengaktifkan setiap penerapan Kubernetes untuk melakukan streaming hanya konfigurasi untuk node virtual App Mesh miliknya sendiri.
  - a. Simpan konten berikut ini ke file bernama `proxy-auth.json` pada komputer Anda. Pastikan untuk mengganti *alternate-colored values* dengan milik Anda sendiri.

JSON

```

{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "appmesh:StreamAggregatedResources",
 "Resource": [
 "arn:aws:appmesh:us-east-1:111122223333:mesh/my-mesh/virtualNode/my-service-a_my-apps"
]
 }
]
}

```

- b. Buat kebijakan.

```
aws iam create-policy --policy-name my-policy --policy-document file://proxy-auth.json
```

- c. Buat peran IAM, lampirkan kebijakan yang Anda buat pada langkah sebelumnya, buat akun layanan Kubernetes, dan ikat kebijakan ke akun layanan Kubernetes. Peran ini memungkinkan pengontrol untuk menambah, menghapus, dan mengubah resource App Mesh.

```
eksctl create iamserviceaccount \
 --cluster $CLUSTER_NAME \
 --namespace my-apps \
 --name my-service-a \
 --attach-policy-arn arn:aws:iam::111122223333:policy/my-policy \
 --override-existing-serviceaccounts \
 --approve
```

Jika Anda lebih suka membuat akun layanan menggunakan Konsol Manajemen AWS atau AWS CLI, lihat [Membuat Peran dan kebijakan IAM untuk akun layanan Anda](#) di Panduan Pengguna Amazon EKS. Jika Anda menggunakan Konsol Manajemen AWS atau AWS CLI untuk membuat akun, Anda juga perlu memetakan peran tersebut ke akun layanan Kubernetes. Untuk informasi selengkapnya, lihat [Menentukan peran IAM untuk akun layanan Anda](#) di Panduan Pengguna Amazon EKS.

2. (Opsional) Jika Anda ingin menerapkan penerapan Anda ke pod Fargate, maka Anda perlu membuat profil Fargate. Jika Anda belum eksctl menginstal, Anda dapat menginstalnya dengan petunjuk di [Instalasi atau Upgrade eksctl](#) di Panduan Pengguna Amazon EKS. Jika Anda lebih suka membuat profil menggunakan konsol, lihat [Membuat profil Fargate](#) di Panduan Pengguna Amazon EKS.

```
eksctl create fargateprofile --cluster my-cluster --region Region-code --name my-service-a --namespace my-apps
```

3. Buat layanan dan deployment Kubernetes. Jika Anda memiliki penerapan yang sudah ada yang ingin Anda gunakan dengan App Mesh, maka Anda perlu menerapkan node virtual, seperti yang Anda lakukan di 3 sub-langkah. [the section called “Langkah 2: Menyebarkan sumber daya App Mesh”](#) Perbarui penerapan Anda untuk memastikan bahwa labelnya cocok dengan label yang Anda tetapkan pada node virtual, sehingga kontainer sidecar ditambahkan secara otomatis ke pod dan pod di-deploy ulang.

- a. Simpan konten berikut ini ke file bernama `example-service.yaml` pada komputer Anda. Jika Anda mengubah nama namespace dan menggunakan pod Fargate, pastikan nama namespace cocok dengan nama namespace yang Anda tentukan di profil Fargate Anda.

```
apiVersion: v1
kind: Service
metadata:
 name: my-service-a
 namespace: my-apps
 labels:
 app: my-app-1
spec:
 selector:
 app: my-app-1
 ports:
 - protocol: TCP
 port: 80
 targetPort: 80

apiVersion: apps/v1
kind: Deployment
metadata:
 name: my-service-a
 namespace: my-apps
 labels:
 app: my-app-1
spec:
 replicas: 3
 selector:
 matchLabels:
 app: my-app-1
 template:
 metadata:
 labels:
 app: my-app-1
 spec:
 serviceAccountName: my-service-a
 containers:
 - name: nginx
 image: nginx:1.19.0
 ports:
 - containerPort: 80
```

**⚠ Important**

Nilai untuk spesifikasi harus sesuai dengan nilai yang Anda tentukan saat Anda membuat node virtual di sub-langkah 3 [the section called “Langkah 2: Menyebarkan sumber daya App Mesh”](#), atau kontainer sidecar tidak akan disuntikkan ke dalam pod. `app matchLabels selector` Pada contoh sebelumnya, nilai untuk label adalah `my-app-1`. Jika Anda menerapkan gateway virtual, bukan node virtual, maka Deployment manifes harus menyertakan hanya wadah Envoy. Untuk informasi selengkapnya tentang gambar yang akan digunakan, lihat [Utusan](#). Untuk contoh manifest, lihat [contoh penerapan](#) di GitHub

- b. Menyebarkan layanan.

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

- c. Lihat layanan dan penyebaran.

```
kubectl -n my-apps get pods
```

## Output

| NAME                                       | READY | STATUS  | RESTARTS | AGE |
|--------------------------------------------|-------|---------|----------|-----|
| <code>my-service-a-54776556f6-2cxd9</code> | 2/2   | Running | 0        | 10s |
| <code>my-service-a-54776556f6-w26kf</code> | 2/2   | Running | 0        | 18s |
| <code>my-service-a-54776556f6-zw5kt</code> | 2/2   | Running | 0        | 26s |

- d. Lihat detail untuk salah satu pod yang di-deploy.

```
kubectl -n my-apps describe pod my-service-a-54776556f6-2cxd9
```

## Output dipersingkat

```
Name: my-service-a-54776556f6-2cxd9
Namespace: my-app-1
Priority: 0
Node: ip-192-168-44-157.us-west-2.compute.internal/192.168.44.157
Start Time: Wed, 17 Jun 2020 11:08:59 -0500
Labels: app=nginx
 pod-template-hash=54776556f6
```

```
Annotations: kubernetes.io/psp: eks.privileged
Status: Running
IP: 192.168.57.134
IPs:
 IP: 192.168.57.134
Controlled By: ReplicaSet/my-service-a-54776556f6
Init Containers:
 proxyinit:
 Container ID: docker://
e0c4810d584c21ae0cb6e40f6119d2508f029094d0e01c9411c6cf2a32d77a59
 Image: 111345817488.dkr.ecr.us-west-2.amazonaws.com/aws-appmesh-
proxy-route-manager:v2
 Image ID: docker-pullable://111345817488.dkr.ecr.us-
west-2.amazonaws.com/aws-appmesh-proxy-route-manager
 Port: <none>
 Host Port: <none>
 State: Terminated
 Reason: Completed
 Exit Code: 0
 Started: Fri, 26 Jun 2020 08:36:22 -0500
 Finished: Fri, 26 Jun 2020 08:36:22 -0500
 Ready: True
 Restart Count: 0
 Requests:
 cpu: 10m
 memory: 32Mi
 Environment:
 APPMESH_START_ENABLED: 1
 APPMESH_IGNORE_UID: 1337
 APPMESH_ENVOY_INGRESS_PORT: 15000
 APPMESH_ENVOY_EGRESS_PORT: 15001
 APPMESH_APP_PORTS: 80
 APPMESH_EGRESS_IGNORED_IP: 169.254.169.254
 APPMESH_EGRESS_IGNORED_PORTS: 22
 AWS_ROLE_ARN: arn:aws:iam::111122223333:role/eksctl-app-
mesh-addon-iamserviceaccount-my-a-Role1-NMNCVWB6PL0N
 AWS_WEB_IDENTITY_TOKEN_FILE: /var/run/secrets/eks.amazonaws.com/
serviceaccount/token
 ...
Containers:
 nginx:
 Container ID: docker://
be6359dc6ecd3f18a1c87df7b57c2093e1f9db17d5b3a77f22585ce3bcab137a
 Image: nginx:1.19.0
```

```

Image ID: docker-pullable://nginx
Port: 80/TCP
Host Port: 0/TCP
State: Running
 Started: Fri, 26 Jun 2020 08:36:28 -0500
Ready: True
Restart Count: 0
Environment:
 AWS_ROLE_ARN: arn:aws:iam::111122223333:role/eksctl-app-
mesh-addon-iamserviceaccount-my-a-Role1-NMNCVWB6PL0N
 AWS_WEB_IDENTITY_TOKEN_FILE: /var/run/secrets/eks.amazonaws.com/
serviceaccount/token
 ...
envoy:
 Container ID:
docker://905b55cbf33ef3b3debc51cb448401d24e2e7c2dbfc6a9754a2c49dd55a216b6
 Image: 840364872350.dkr.ecr.us-west-2.amazonaws.com/aws-appmesh-
envoy:v1.12.4.0-prod
 Image ID: docker-pullable://840364872350.dkr.ecr.us-
west-2.amazonaws.com/aws-appmesh-envoy
 Port: 9901/TCP
 Host Port: 0/TCP
 State: Running
 Started: Fri, 26 Jun 2020 08:36:36 -0500
Ready: True
Restart Count: 0
Requests:
 cpu: 10m
 memory: 32Mi
Environment:
 APPMESH_RESOURCE_ARN: arn:aws:iam::111122223333:mesh/my-mesh/
virtualNode/my-service-a_my-apps
 APPMESH_PREVIEW: 0
 ENVOY_LOG_LEVEL: info
 AWS_REGION: us-west-2
 AWS_ROLE_ARN: arn:aws:iam::111122223333:role/eksctl-app-
mesh-addon-iamserviceaccount-my-a-Role1-NMNCVWB6PL0N
 AWS_WEB_IDENTITY_TOKEN_FILE: /var/run/secrets/eks.amazonaws.com/
serviceaccount/token
 ...
Events:
 Type Reason Age From
 Message

```


```

Normal Pulling 30s kubelet, ip-192-168-44-157.us-
west-2.compute.internal Pulling image "111345817488.dkr.ecr.us-
west-2.amazonaws.com/aws-appmesh-proxy-route-manager:v2"
Normal Pulled 23s kubelet, ip-192-168-44-157.us-
west-2.compute.internal Successfully pulled image "111345817488.dkr.ecr.us-
west-2.amazonaws.com/aws-appmesh-proxy-route-manager:v2"
Normal Created 21s kubelet, ip-192-168-44-157.us-
west-2.compute.internal Created container proxyinit
Normal Started 21s kubelet, ip-192-168-44-157.us-
west-2.compute.internal Started container proxyinit
Normal Pulling 20s kubelet, ip-192-168-44-157.us-
west-2.compute.internal Pulling image "nginx:1.19.0"
Normal Pulled 16s kubelet, ip-192-168-44-157.us-
west-2.compute.internal Successfully pulled image "nginx:1.19.0"
Normal Created 15s kubelet, ip-192-168-44-157.us-
west-2.compute.internal Created container nginx
Normal Started 15s kubelet, ip-192-168-44-157.us-
west-2.compute.internal Started container nginx
Normal Pulling 15s kubelet, ip-192-168-44-157.us-
west-2.compute.internal Pulling image "840364872350.dkr.ecr.us-
west-2.amazonaws.com/aws-appmesh-envoy:v1.12.4.0-prod"
Normal Pulled 8s kubelet, ip-192-168-44-157.us-
west-2.compute.internal Successfully pulled image "840364872350.dkr.ecr.us-
west-2.amazonaws.com/aws-appmesh-envoy:v1.12.4.0-prod"
Normal Created 7s kubelet, ip-192-168-44-157.us-
west-2.compute.internal Created container envoy
Normal Started 7s kubelet, ip-192-168-44-157.us-
west-2.compute.internal Started container envoy

```

Pada output sebelumnya, Anda dapat melihat bahwa envoy kontainer proxyinit dan ditambahkan ke pod oleh pengontrol. Jika Anda menerapkan layanan contoh ke Fargate, maka envoy kontainer ditambahkan ke pod oleh pengontrol, tetapi proxyinit kontainer tidak.

4. (Opsional) Instal add-on seperti Prometheus, Grafana, Jaeger, dan Datadog. AWS X-Ray Untuk informasi selengkapnya, lihat [add-on App Mesh](#) GitHub dan bagian [Observabilitas](#) dari Panduan Pengguna App Mesh.

 Note

Untuk contoh dan penelusuran lainnya untuk App Mesh, lihat repositori contoh [App Mesh](#).

## Langkah 4: Membersihkan

Hapus semua sumber daya contoh yang dibuat dalam tutorial ini. Pengontrol juga menghapus sumber daya yang dibuat di mesh layanan `my-mesh` App Mesh.

```
kubectl delete namespace my-apps
```

Jika Anda membuat profil Fargate untuk layanan contoh, maka hapus.

```
eksctl delete fargateprofile --name my-service-a --cluster my-cluster --region Region-code
```

Hapus jala.

```
kubectl delete mesh my-mesh
```


(Opsional) Anda dapat menghapus komponen integrasi Kubernetes.

```
helm delete appmesh-controller -n appmesh-system
```

(Opsional) Jika Anda menerapkan komponen integrasi Kubernetes ke Fargate, hapus profil Fargate.

```
eksctl delete fargateprofile --name appmesh-system --cluster my-cluster --region Region-code
```

## Memulai dengan AWS App Mesh dan Amazon EC2

 Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk. AWS App Mesh Setelah 30 September 2026, Anda tidak akan lagi dapat

mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Topik ini membantu Anda menggunakan AWS App Mesh layanan aktual yang berjalan di Amazon EC2. Tutorial ini mencakup fitur dasar dari beberapa jenis sumber daya App Mesh.

## Skenario

Untuk mengilustrasikan cara menggunakan App Mesh, asumsikan bahwa Anda memiliki aplikasi dengan karakteristik sebagai berikut:

- Terdiri dari dua layanan bernama `serviceA` dan `serviceB`.
- Kedua layanan terdaftar ke namespace bernama `apps.local`
- `ServiceA` berkomunikasi dengan `serviceB` lebih dari HTTP/2, port 80.
- Anda telah menerapkan versi 2 `serviceB` dan mendaftarkannya dengan nama `serviceBv2` di `apps.local` namespace.

Anda memiliki persyaratan berikut:

- Anda ingin mengirim 75 persen lalu lintas dari `serviceA` ke `serviceB` dan 25 persen lalu lintas ke yang `serviceBv2` pertama. Dengan hanya mengirim 25 persen ke `serviceBv2`, Anda dapat memvalidasi bahwa itu bebas bug sebelum Anda mengirim 100 persen lalu lintas dari `serviceA`.
- Anda ingin dapat dengan mudah menyesuaikan bobot lalu lintas sehingga 100 persen lalu lintas masuk `serviceBv2` setelah terbukti dapat diandalkan. Setelah semua lalu lintas dikirim ke `serviceBv2`, Anda ingin berhent `serviceB`.
- Anda tidak ingin harus mengubah kode aplikasi atau pendaftaran penemuan layanan yang ada untuk layanan Anda yang sebenarnya untuk memenuhi persyaratan sebelumnya.

Untuk memenuhi kebutuhan Anda, Anda memutuskan untuk membuat mesh layanan App Mesh dengan layanan virtual, node virtual, router virtual, dan rute. Setelah menerapkan mesh Anda, Anda memperbarui layanan Anda untuk menggunakan proxy Envoy. Setelah diperbarui, layanan Anda berkomunikasi satu sama lain melalui proxy Utusan daripada langsung satu sama lain.

## Prasyarat

App Mesh mendukung layanan Linux yang terdaftar dengan DNS, AWS Cloud Map, atau keduanya. Untuk menggunakan panduan memulai ini, kami sarankan Anda memiliki tiga layanan yang sudah ada yang terdaftar di DNS. Anda dapat membuat mesh layanan dan sumber dayanya bahkan jika layanan tidak ada, tetapi Anda tidak dapat menggunakan mesh sampai Anda telah menerapkan layanan yang sebenarnya.

Jika Anda belum menjalankan layanan, Anda dapat meluncurkan instans Amazon EC2 dan menyebarkan aplikasi ke dalamnya. Untuk informasi selengkapnya, lihat [Tutorial: Memulai instans Amazon EC2 Linux di Panduan Pengguna Amazon EC2](#). Langkah-langkah yang tersisa mengasumsikan bahwa layanan yang sebenarnya diberi nama `serviceA`, `serviceB`, `serviceC`, dan bahwa semua layanan dapat ditemukan melalui namespace bernama `apps.local`.

## Langkah 1: Buat mesh dan layanan virtual

Mesh layanan adalah batas logis untuk lalu lintas jaringan antara layanan yang berada di dalamnya. Untuk informasi selengkapnya, lihat [Jala Layanan](#). Layanan virtual adalah abstraksi dari layanan yang sebenarnya. Untuk informasi selengkapnya, lihat [Layanan virtual](#).

Buat sumber daya berikut:

- Sebuah mesh bernama `apps`, karena semua layanan dalam skenario terdaftar ke `apps.local` namespace.
- Layanan virtual bernama `serviceb.apps.local`, karena layanan virtual mewakili layanan yang dapat ditemukan dengan nama itu, dan Anda tidak ingin mengubah kode Anda untuk merujuk nama lain. Layanan virtual bernama `servicea.apps.local` ditambahkan pada langkah selanjutnya.

Anda dapat menggunakan AWS CLI versi 1.18.116 Konsol Manajemen AWS atau lebih tinggi atau 2.0.38 atau lebih tinggi untuk menyelesaikan langkah-langkah berikut. Jika menggunakan AWS CLI, gunakan `aws --version` perintah untuk memeriksa AWS CLI versi yang Anda instal. [Jika Anda tidak memiliki versi 1.18.116 atau lebih tinggi atau 2.0.38 atau lebih tinggi diinstal, maka Anda harus menginstal atau memperbarui. AWS CLI](#) Pilih tab untuk alat yang ingin Anda gunakan.

### Konsol Manajemen AWS

1. Buka wizard yang dijalankan pertama kali konsol App Mesh saat memulai <https://console.aws.amazon.com/appmesh/>.

2. Untuk nama Mesh, masukkan **apps**.
3. Untuk nama layanan virtual, masukkan **serviceb.apps.local**.
4. Untuk melanjutkan, pilih Berikutnya.

## AWS CLI

1. Buat mesh dengan [create-mesh](#) perintah.

```
aws appmesh create-mesh --mesh-name apps
```

2. Buat layanan virtual dengan [create-virtual-service](#) perintah.

```
aws appmesh create-virtual-service --mesh-name apps --virtual-service-name serviceb.apps.local --spec {}
```

## Langkah 2: Buat simpul virtual

Node virtual bertindak sebagai pointer logis ke layanan yang sebenarnya. Untuk informasi selengkapnya, lihat [Node virtual](#).

Buat node virtual bernama `serviceB`, karena salah satu node virtual mewakili layanan yang sebenarnya bernama `serviceB`. Layanan aktual yang diwakili oleh node virtual dapat ditemukan melalui DNS dengan nama host dari `serviceb.apps.local`. Sebagai alternatif, Anda dapat menemukan layanan aktual menggunakan AWS Cloud Map. Node virtual mendengarkan lalu lintas menggunakan protokol HTTP/2 pada port 80. Protokol lain juga didukung, seperti halnya pemeriksaan kesehatan. Anda membuat node virtual untuk `serviceA` dan `serviceBv2` di langkah selanjutnya.

### Konsol Manajemen AWS

1. Untuk nama simpul Virtual, masukkan **serviceB**.
2. Untuk metode Penemuan layanan, pilih DNS dan masukkan **serviceb.apps.local** untuk nama host DNS.
3. Di bawah konfigurasi Listener, pilih `http2` untuk Protokol dan masukkan **80** untuk Port.
4. Untuk melanjutkan, pilih Berikutnya.

## AWS CLI

1. Buat file bernama `create-virtual-node-serviceb.json` dengan konten berikut:

```
{
 "meshName": "apps",
 "spec": {
 "listeners": [
 {
 "portMapping": {
 "port": 80,
 "protocol": "http2"
 }
 }
],
 "serviceDiscovery": {
 "dns": {
 "hostname": "serviceB.apps.local"
 }
 }
 },
 "virtualNodeName": "serviceB"
}
```

2. Buat node virtual dengan [create-virtual-node](#) perintah menggunakan file JSON sebagai input.

```
aws appmesh create-virtual-node --cli-input-json file://create-virtual-node-serviceb.json
```

## Langkah 3: Buat router virtual dan rute

Router virtual merutekan lalu lintas untuk satu atau lebih layanan virtual di dalam mesh Anda. Untuk informasi selengkapnya, lihat [Router virtual](#) dan [Rute](#).

Buat sumber daya berikut:

- Router virtual bernama `serviceB`, karena layanan `serviceB.apps.local` virtual tidak memulai komunikasi keluar dengan layanan lain. Ingatlah bahwa layanan virtual yang Anda buat sebelumnya adalah abstraksi dari `serviceb.apps.local` layanan Anda yang sebenarnya. Layanan virtual mengirimkan lalu lintas ke router virtual. Router virtual mendengarkan lalu lintas menggunakan protokol HTTP/2 pada port 80. Protokol lain juga didukung.

- Sebuah rute bernama `serviceB`. Ini merutekan 100 persen lalu lintasnya ke node `serviceB` virtual. Bobotnya ada di langkah selanjutnya setelah Anda menambahkan simpul `serviceBv2` virtual. Meskipun tidak tercakup dalam panduan ini, Anda dapat menambahkan kriteria filter tambahan untuk rute dan menambahkan kebijakan coba lagi untuk menyebabkan proxy Envoy melakukan beberapa upaya untuk mengirim lalu lintas ke node virtual ketika mengalami masalah komunikasi.

## Konsol Manajemen AWS

1. Untuk nama router Virtual, masukkan `serviceB`.
2. Di bawah konfigurasi Listener, pilih `http2` untuk Protokol dan tentukan `80` untuk Port.
3. Untuk nama Rute, masukkan `serviceB`.
4. Untuk jenis Rute, pilih `http2`.
5. Untuk nama simpul Virtual di bawah konfigurasi Target, pilih `serviceB` dan masukkan `100` untuk Berat.
6. Di bawah konfigurasi Match, pilih Metode.
7. Untuk melanjutkan, pilih Berikutnya.

## AWS CLI

1. Buat router virtual.
  - a. Buat file bernama `create-virtual-router.json` dengan konten berikut:

```
{
 "meshName": "apps",
 "spec": {
 "listeners": [
 {
 "portMapping": {
 "port": 80,
 "protocol": "http2"
 }
 }
]
 },
 "virtualRouterName": "serviceB"
}
```

```
}
```

- b. Buat router virtual dengan [create-virtual-router](#) perintah menggunakan file JSON sebagai input.

```
aws appmesh create-virtual-router --cli-input-json file://create-virtual-router.json
```

## 2. Buat rute.

- a. Buat file bernama `create-route.json` dengan konten berikut:

```
{
 "meshName" : "apps",
 "routeName" : "serviceB",
 "spec" : {
 "httpRoute" : {
 "action" : {
 "weightedTargets" : [
 {
 "virtualNode" : "serviceB",
 "weight" : 100
 }
]
 },
 "match" : {
 "prefix" : "/"
 }
 }
 },
 "virtualRouterName" : "serviceB"
}
```

- b. Buat rute dengan perintah [create-route](#) menggunakan file JSON sebagai input.

```
aws appmesh create-route --cli-input-json file://create-route.json
```

## Langkah 4: Tinjau dan buat

Tinjau pengaturan terhadap instruksi sebelumnya.

## Konsol Manajemen AWS

Pilih Edit jika Anda perlu membuat perubahan di bagian mana pun. Setelah Anda puas dengan pengaturan, pilih Buat mesh.

Layar Status menunjukkan kepada Anda semua sumber daya mesh yang dibuat. Anda dapat melihat sumber daya yang dibuat di konsol dengan memilih View mesh.

## AWS CLI

Tinjau pengaturan mesh yang Anda buat dengan [perintah describe-mesh](#).

```
aws appmesh describe-mesh --mesh-name apps
```

Tinjau pengaturan layanan virtual yang Anda buat dengan [describe-virtual-service](#) perintah.

```
aws appmesh describe-virtual-service --mesh-name apps --virtual-service-name
serviceb.apps.local
```

Tinjau pengaturan node virtual yang Anda buat dengan [describe-virtual-node](#) perintah.

```
aws appmesh describe-virtual-node --mesh-name apps --virtual-node-name serviceB
```

Tinjau pengaturan router virtual yang Anda buat dengan [describe-virtual-router](#) perintah.

```
aws appmesh describe-virtual-router --mesh-name apps --virtual-router-name serviceB
```

Tinjau pengaturan rute yang Anda buat dengan [perintah deskripsi-rute](#).

```
aws appmesh describe-route --mesh-name apps \
--virtual-router-name serviceB --route-name serviceB
```

## Langkah 5: Buat sumber daya tambahan

Untuk menyelesaikan skenario, Anda perlu:

- Buat satu node virtual bernama `serviceBv2` dan yang lain bernama `serviceA`. Kedua node virtual mendengarkan permintaan melalui HTTP/2 port 80. Untuk node `serviceA` virtual, konfigurasi backend dari `serviceb.apps.local` Semua lalu lintas keluar dari node

`serviceA` virtual dikirim ke layanan virtual bernama `serviceb.apps.local`. Meskipun tidak tercakup dalam panduan ini, Anda juga dapat menentukan jalur file untuk menulis log akses untuk node virtual.

- Buat satu layanan virtual tambahan bernama `servicea.apps.local`, yang mengirimkan semua lalu lintas langsung ke node `serviceA` virtual.
- Perbarui `serviceB` rute yang Anda buat pada langkah sebelumnya untuk mengirim 75 persen lalu lintasnya ke node `serviceB` virtual dan 25 persen lalu lintasnya ke node `serviceBv2` virtual. Seiring waktu, Anda dapat terus memodifikasi bobot hingga `serviceBv2` menerima 100 persen dari lalu lintas. Setelah semua lalu lintas dikirim ke `serviceBv2`, Anda dapat mematikan dan menghentikan node `serviceB` virtual dan layanan aktual. Saat Anda mengubah bobot, kode Anda tidak memerlukan modifikasi apa pun, karena nama layanan `serviceb.apps.local` virtual dan aktual tidak berubah. Ingatlah bahwa layanan `serviceb.apps.local` virtual mengirimkan lalu lintas ke router virtual, yang merutekan lalu lintas ke node virtual. Nama penemuan layanan untuk node virtual dapat diubah kapan saja.

## Konsol Manajemen AWS

1. Di panel navigasi kiri, pilih Meshes.
2. Pilih apps mesh yang Anda buat pada langkah sebelumnya.
3. Di panel navigasi kiri, pilih Virtual nodes.
4. Pilih Buat simpul virtual.
5. Untuk nama simpul Virtual, masukkan **serviceBv2**, untuk metode penemuan Layanan, pilih DNS, dan untuk nama host DNS, masukkan. **servicebv2.apps.local**
6. Untuk konfigurasi Listener, pilih http2 untuk Protokol dan masukkan **80** untuk Port.
7. Pilih Buat simpul virtual.
8. Pilih Buat simpul virtual lagi. Masukkan **serviceA** untuk nama simpul Virtual. Untuk metode Penemuan layanan, pilih DNS, dan untuk nama host DNS, masukkan. **servicea.apps.local**
9. Untuk Masukkan nama layanan virtual di bawah Backend baru, masukkan. **serviceb.apps.local**
10. Di bawah konfigurasi Listener, pilih http2 untuk Protokol, masukkan **80** untuk Port, dan kemudian pilih Buat simpul virtual.
11. Di panel navigasi kiri, pilih Router virtual dan kemudian pilih router `serviceB` virtual dari daftar.

12. Di bawah Rute, pilih rute bernama **ServiceB** yang Anda buat pada langkah sebelumnya, dan pilih Edit.
13. Di bawah Target, nama simpul Virtual, ubah nilai Weight serviceB untuk menjadi **75**.
14. Pilih Tambah target, pilih serviceBv2 dari daftar dropdown, dan atur nilai Weight ke. **25**
15. Pilih Simpan.
16. Di panel navigasi kiri, pilih Layanan virtual dan kemudian pilih Buat layanan virtual.
17. Masukkan **servicea.apps.local** nama layanan Virtual, pilih Virtual node for Provider, pilih serviceA untuk Virtual node, dan kemudian pilih Create Virtual Service.

## AWS CLI

1. Buat node serviceBv2 virtual.
  - a. Buat file bernama `create-virtual-node-servicebv2.json` dengan konten berikut:

```
{
 "meshName": "apps",
 "spec": {
 "listeners": [
 {
 "portMapping": {
 "port": 80,
 "protocol": "http2"
 }
 }
],
 "serviceDiscovery": {
 "dns": {
 "hostname": "serviceBv2.apps.local"
 }
 }
 },
 "virtualNodeName": "serviceBv2"
}
```

- b. Buat node virtual.

```
aws appmesh create-virtual-node --cli-input-json file://create-virtual-node-servicebv2.json
```

2. Buat node serviceA virtual.

a. Buat file bernama `create-virtual-node-servicea.json` dengan konten berikut:

```
{
 "meshName" : "apps",
 "spec" : {
 "backends" : [
 {
 "virtualService" : {
 "virtualServiceName" : "serviceb.apps.local"
 }
 }
],
 "listeners" : [
 {
 "portMapping" : {
 "port" : 80,
 "protocol" : "http2"
 }
 }
],
 "serviceDiscovery" : {
 "dns" : {
 "hostname" : "servicea.apps.local"
 }
 }
 },
 "virtualNodeName" : "serviceA"
}
```

b. Buat node virtual.

```
aws appmesh create-virtual-node --cli-input-json file://create-virtual-node-servicea.json
```

3. Perbarui layanan `serviceb.apps.local` virtual yang Anda buat pada langkah sebelumnya untuk mengirim lalu lintas ke router `serviceB` virtual. Ketika layanan virtual awalnya dibuat, itu tidak mengirim lalu lintas ke mana pun, karena router `serviceB` virtual belum dibuat.

- a. Buat file bernama `update-virtual-service.json` dengan konten berikut:

```
{
 "meshName" : "apps",
 "spec" : {
 "provider" : {
 "virtualRouter" : {
 "virtualRouterName" : "serviceB"
 }
 }
 },
 "virtualServiceName" : "serviceb.apps.local"
}
```

- b. Perbarui layanan virtual dengan [update-virtual-service](#) perintah.

```
aws appmesh update-virtual-service --cli-input-json file://update-virtual-
service.json
```

4. Perbarui `serviceB` rute yang Anda buat di langkah sebelumnya.

- a. Buat file bernama `update-route.json` dengan konten berikut:

```
{
 "meshName" : "apps",
 "routeName" : "serviceB",
 "spec" : {
 "http2Route" : {
 "action" : {
 "weightedTargets" : [
 {
 "virtualNode" : "serviceB",
 "weight" : 75
 },
 {
 "virtualNode" : "serviceBv2",
 "weight" : 25
 }
]
 },
 "match" : {
 "prefix" : "/"
 }
 }
 }
}
```

```
 }
 }
},
"virtualRouterName" : "serviceB"
}
```

- b. Perbarui rute dengan perintah [update-route](#).

```
aws appmesh update-route --cli-input-json file://update-route.json
```

5. Buat layanan serviceA virtual.

- a. Buat file bernama `create-virtual-servicea.json` dengan konten berikut:

```
{
 "meshName" : "apps",
 "spec" : {
 "provider" : {
 "virtualNode" : {
 "virtualNodeName" : "serviceA"
 }
 }
 },
 "virtualServiceName" : "servicea.apps.local"
}
```

- b. Buat layanan virtual.

```
aws appmesh create-virtual-service --cli-input-json file://create-virtual-servicea.json
```

## Ringkasan mesh

Sebelum Anda membuat mesh layanan, Anda memiliki tiga layanan aktual bernama `servicea.apps.local`, `serviceb.apps.local`, dan `servicebv2.apps.local`. Selain layanan yang sebenarnya, Anda sekarang memiliki mesh layanan yang berisi sumber daya berikut yang mewakili layanan sebenarnya:

- Dua layanan virtual. Proxy mengirimkan semua lalu lintas dari layanan `servicea.apps.local` virtual ke layanan `serviceb.apps.local` virtual melalui router virtual.

- Tiga node virtual bernama `serviceA`, `serviceB`, dan `serviceBv2`. Proxy Envoy menggunakan informasi penemuan layanan yang dikonfigurasi untuk node virtual untuk mencari alamat IP dari layanan yang sebenarnya.
- Satu router virtual dengan satu rute yang menginstruksikan proxy Utusan untuk merutekan 75 persen lalu lintas masuk ke node `serviceB` virtual dan 25 persen lalu lintas ke node virtual `serviceBv2`.

## Langkah 6: Perbarui layanan

Setelah membuat mesh Anda, Anda harus menyelesaikan tugas-tugas berikut:

- Otorisasi proxy Envoy yang Anda gunakan dengan setiap layanan untuk membaca konfigurasi satu atau lebih node virtual. Untuk informasi selengkapnya tentang cara mengotorisasi proxy, lihat [Otorisasi Proxy Utusan](#).
- Untuk memperbarui layanan Anda yang ada, selesaikan langkah-langkah berikut.

Untuk mengonfigurasi instans Amazon EC2 sebagai anggota node virtual

1. Buat peran IAM.
  - a. Buat file bernama `ec2-trust-relationship.json` dengan isi berikut ini.

JSON

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

- b. Buat peran IAM dengan perintah berikut.

```
aws iam create-role --role-name mesh-virtual-node-service-b --assume-role-policy-document file://ec2-trust-relationship.json
```

2. Lampirkan kebijakan IAM ke peran yang memungkinkannya membaca dari Amazon ECR dan hanya konfigurasi node virtual App Mesh tertentu.
  - a. Buat file bernama `virtual-node-policy.json` dengan konten berikut. apps adalah nama mesh yang Anda buat [the section called “Langkah 1: Buat mesh dan layanan virtual”](#) dan `serviceB` merupakan nama node virtual yang Anda buat [the section called “Langkah 2: Buat simpul virtual”](#). Ganti `111122223333` dengan ID akun Anda dan `us-west-2` dengan Wilayah tempat Anda membuat mesh.

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "appmesh:StreamAggregatedResources",
 "Resource": [
 "arn:aws:appmesh:us-west-2:111122223333:mesh/apps/virtualNode/serviceB"
]
 }
]
}
```

- b. Buat kebijakan dengan perintah berikut.

```
aws iam create-policy --policy-name virtual-node-policy --policy-document file://virtual-node-policy.json
```

- c. Lampirkan kebijakan yang Anda buat di langkah sebelumnya ke peran sehingga peran dapat membaca konfigurasi hanya untuk node `serviceB` virtual dari App Mesh.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::111122223333:policy/virtual-node-policy --role-name mesh-virtual-node-service-b
```

- d. Lampirkan kebijakan AmazonEC2ContainerRegistryReadOnly terkelola ke peran sehingga dapat menarik image container Envoy dari Amazon ECR.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEC2ContainerRegistryReadOnly --role-name mesh-virtual-node-service-b
```

3. [Luncurkan instans Amazon EC2 dengan peran IAM](#) yang Anda buat.
4. Connect ke instans Anda melalui SSH.
5. Instal Docker dan AWS CLI pada instance Anda sesuai dengan dokumentasi sistem operasi Anda.
6. Otentikasi ke repositori Envoy Amazon ECR di Wilayah tempat Anda ingin klien Docker Anda menarik gambarnya.
  - Semua Wilayah kecuallime-south-1,ap-east-1,ap-southeast-3,eu-south-1,il-central-1, danaf-south-1. Anda dapat mengganti *us-west-2* dengan [Wilayah yang didukung](#) kecuallime-south-1,,ap-east-1,ap-southeast-3,eu-south-1,il-central-1, danaf-south-1.

```
$aws ecr get-login-password \
 --region us-west-2 \
| docker login \
 --username AWS \
 --password-stdin 840364872350.dkr.ecr.us-west-2.amazonaws.com
```

- Wilayah me-south-1

```
$aws ecr get-login-password \
 --region me-south-1 \
| docker login \
 --username AWS \
 --password-stdin 772975370895.dkr.ecr.me-south-1.amazonaws.com
```

- Wilayah ap-east-1

```
$aws ecr get-login-password \
 --region ap-east-1 \
| docker login \
 --username AWS \
 --password-stdin 856666278305.dkr.ecr.ap-east-1.amazonaws.com
```

7. Jalankan salah satu perintah berikut untuk memulai container App Mesh Envoy pada instance Anda, bergantung pada Region mana Anda ingin menarik gambarnya. *serviceB* Nilai-nilai *apps* dan adalah mesh dan nama node virtual yang didefinisikan dalam skenario. Informasi ini memberi tahu proxy konfigurasi node virtual mana yang akan dibaca dari App Mesh. Untuk menyelesaikan skenario, Anda juga perlu menyelesaikan langkah-langkah ini untuk instans Amazon EC2 yang meng-host layanan yang diwakili oleh node *serviceBv2* dan *serviceA* virtual. Untuk aplikasi Anda sendiri, ganti nilai-nilai ini dengan milik Anda sendiri.
- Semua Wilayah *kecualime-south-1,ap-east-1,ap-southeast-3,eu-south-1,il-central-1, dan af-south-1*. Anda dapat mengganti *Region-code* dengan [Wilayah apa pun yang didukung](#) *kecualime-south-1,ap-east-1,ap-southeast-3,eu-south-1,,il-central-1, dan af-south-1* Wilayah. Anda dapat mengganti *1337* dengan nilai apa pun antara *0* dan *2147483647*.

```
sudo docker run --detach --env APPMESH_RESOURCE_ARN=mesh/apps/
virtualNode/serviceB \
-u 1337 --network host 840364872350.dkr.ecr.region-code.amazonaws.com/aws-
appmesh-envoy:v1.34.13.0-prod
```

- me-south-1* Wilayah. Anda dapat mengganti *1337* dengan nilai apa pun antara *0* dan *2147483647*.

```
sudo docker run --detach --env APPMESH_RESOURCE_ARN=mesh/apps/
virtualNode/serviceB \
-u 1337 --network host 772975370895.dkr.ecr.me-south-1.amazonaws.com/aws-
appmesh-
envoy:v1.34.13.0-prod
```

- ap-east-1* Wilayah. Anda dapat mengganti *1337* dengan nilai apa pun antara *0* dan *2147483647*.

```
sudo docker run --detach --env APPMESH_RESOURCE_ARN=mesh/apps/
virtualNode/serviceB \
-u 1337 --network host 856666278305.dkr.ecr.ap-east-1.amazonaws.com/aws-
appmesh-
envoy:v1.34.13.0-prod
```

**Note**

APPMESH\_RESOURCE\_ARN Properti memerlukan versi 1.15.0 atau yang lebih baru dari gambar Utusan. Untuk informasi selengkapnya, lihat [Utusan](#).

**Important**

Hanya versi v1.9.0.0-prod atau yang lebih baru yang didukung untuk digunakan dengan App Mesh.

- Pilih **Show more** di bawah ini. Buat file bernama `envoy-networking.sh` pada instance Anda dengan konten berikut. Ganti `8000` dengan port yang digunakan kode aplikasi Anda untuk lalu lintas masuk. Anda dapat mengubah nilainya `APPMESH_IGNORE_UID`, tetapi nilainya harus sama dengan nilai yang Anda tentukan pada langkah sebelumnya; misalnya `1337`. Anda dapat menambahkan alamat tambahan `APPMESH_EGRESS_IGNORED_IP` jika perlu. Jangan memodifikasi baris lainnya.

```
#!/bin/bash -e

#
Start of configurable options
#

#APPMESH_START_ENABLED="0"
APPMESH_IGNORE_UID="1337"
APPMESH_APP_PORTS="8000"
APPMESH_ENVOY_EGRESS_PORT="15001"
APPMESH_ENVOY_INGRESS_PORT="15000"
APPMESH_EGRESS_IGNORED_IP="169.254.169.254,169.254.170.2"

Enable routing on the application start.
[-z "$APPMESH_START_ENABLED"] && APPMESH_START_ENABLED="0"

Enable IPv6.
[-z "$APPMESH_ENABLE_IPV6"] && APPMESH_ENABLE_IPV6="0"
```

```
Egress traffic from the processess owned by the following UID/GID will be
ignored.
if [-z "$APPMESH_IGNORE_UID"] && [-z "$APPMESH_IGNORE_GID"]; then
 echo "Variables APPMESH_IGNORE_UID and/or APPMESH_IGNORE_GID must be set."
 echo "Envoy must run under those IDs to be able to properly route it's egress
traffic."
 exit 1
fi

Port numbers Application and Envoy are listening on.
if [-z "$APPMESH_ENVOY_EGRESS_PORT"]; then
 echo "APPMESH_ENVOY_EGRESS_PORT must be defined to forward traffic from the
application to the proxy."
 exit 1
fi

If an app port was specified, then we also need to enforce the proxies ingress
port so we know where to forward traffic.
if [! -z "$APPMESH_APP_PORTS"] && [-z "$APPMESH_ENVOY_INGRESS_PORT"]; then
 echo "APPMESH_ENVOY_INGRESS_PORT must be defined to forward traffic from the
APPMESH_APP_PORTS to the proxy."
 exit 1
fi

Comma separated list of ports for which egress traffic will be ignored, we always
refuse to route SSH traffic.
if [-z "$APPMESH_EGRESS_IGNORED_PORTS"]; then
 APPMESH_EGRESS_IGNORED_PORTS="22"
else
 APPMESH_EGRESS_IGNORED_PORTS="$APPMESH_EGRESS_IGNORED_PORTS,22"
fi

#
End of configurable options
#

function initialize() {
 echo "=== Initializing ==="
 if [! -z "$APPMESH_APP_PORTS"]; then
 iptables -t nat -N APPMESH_INGRESS
 if ["$APPMESH_ENABLE_IPV6" == "1"]; then
 ip6tables -t nat -N APPMESH_INGRESS
 fi
 fi
fi
```

```
iptables -t nat -N APPMESH_EGRESS
if ["$APPMESH_ENABLE_IPV6" == "1"]; then
 ip6tables -t nat -N APPMESH_EGRESS
fi
}

function enable_egress_routing() {
 # Stuff to ignore
 [! -z "$APPMESH_IGNORE_UID"] && \
 iptables -t nat -A APPMESH_EGRESS \
 -m owner --uid-owner $APPMESH_IGNORE_UID \
 -j RETURN

 [! -z "$APPMESH_IGNORE_GID"] && \
 iptables -t nat -A APPMESH_EGRESS \
 -m owner --gid-owner $APPMESH_IGNORE_GID \
 -j RETURN

 [! -z "$APPMESH_EGRESS_IGNORED_PORTS"] && \
 for IGNORED_PORT in $(echo "$APPMESH_EGRESS_IGNORED_PORTS" | tr "," "\n");
do
 iptables -t nat -A APPMESH_EGRESS \
 -p tcp \
 -m multiport --dports "$IGNORED_PORT" \
 -j RETURN
done

if ["$APPMESH_ENABLE_IPV6" == "1"]; then
 # Stuff to ignore ipv6
 [! -z "$APPMESH_IGNORE_UID"] && \
 ip6tables -t nat -A APPMESH_EGRESS \
 -m owner --uid-owner $APPMESH_IGNORE_UID \
 -j RETURN

 [! -z "$APPMESH_IGNORE_GID"] && \
 ip6tables -t nat -A APPMESH_EGRESS \
 -m owner --gid-owner $APPMESH_IGNORE_GID \
 -j RETURN

 [! -z "$APPMESH_EGRESS_IGNORED_PORTS"] && \
 for IGNORED_PORT in $(echo "$APPMESH_EGRESS_IGNORED_PORTS" | tr "," "\n");
do
 ip6tables -t nat -A APPMESH_EGRESS \
 -p tcp \
```

```

 -m multiport --dports "$IGNORED_PORT" \
 -j RETURN
 done
fi

The list can contain both IPv4 and IPv6 addresses. We will loop over this
list
to add every IPv4 address into `iptables` and every IPv6 address into
`ip6tables`.
[! -z "$APPMESH_EGRESS_IGNORED_IP"] && \
for IP_ADDR in $(echo "$APPMESH_EGRESS_IGNORED_IP" | tr "," "\n"); do
 if [[$IP_ADDR =~ .*:.*]]
 then
 ["$APPMESH_ENABLE_IPV6" == "1"] && \
 ip6tables -t nat -A APPMESH_EGRESS \
 -p tcp \
 -d "$IP_ADDR" \
 -j RETURN
 else
 iptables -t nat -A APPMESH_EGRESS \
 -p tcp \
 -d "$IP_ADDR" \
 -j RETURN
 fi
done

Redirect everything that is not ignored
iptables -t nat -A APPMESH_EGRESS \
 -p tcp \
 -j REDIRECT --to $APPMESH_ENVOY_EGRESS_PORT

Apply APPMESH_EGRESS chain to non local traffic
iptables -t nat -A OUTPUT \
 -p tcp \
 -m addrtype ! --dst-type LOCAL \
 -j APPMESH_EGRESS

if ["$APPMESH_ENABLE_IPV6" == "1"]; then
 # Redirect everything that is not ignored ipv6
 ip6tables -t nat -A APPMESH_EGRESS \
 -p tcp \
 -j REDIRECT --to $APPMESH_ENVOY_EGRESS_PORT
 # Apply APPMESH_EGRESS chain to non local traffic ipv6
 ip6tables -t nat -A OUTPUT \

```

```
 -p tcp \
 -m addrtype ! --dst-type LOCAL \
 -j APPMESH_EGRESS
 fi
}

function enable_ingress_redirect_routing() {
 # Route everything arriving at the application port to Envoy
 iptables -t nat -A APPMESH_INGRESS \
 -p tcp \
 -m multiport --dports "$APPMESH_APP_PORTS" \
 -j REDIRECT --to-port "$APPMESH_ENVOY_INGRESS_PORT"

 # Apply AppMesh ingress chain to everything non-local
 iptables -t nat -A PREROUTING \
 -p tcp \
 -m addrtype ! --src-type LOCAL \
 -j APPMESH_INGRESS

 if ["$APPMESH_ENABLE_IPV6" == "1"]; then
 # Route everything arriving at the application port to Envoy ipv6
 ip6tables -t nat -A APPMESH_INGRESS \
 -p tcp \
 -m multiport --dports "$APPMESH_APP_PORTS" \
 -j REDIRECT --to-port "$APPMESH_ENVOY_INGRESS_PORT"

 # Apply AppMesh ingress chain to everything non-local ipv6
 ip6tables -t nat -A PREROUTING \
 -p tcp \
 -m addrtype ! --src-type LOCAL \
 -j APPMESH_INGRESS
 fi
}

function enable_routing() {
 echo "=== Enabling routing ==="
 enable_egress_routing
 if [! -z "$APPMESH_APP_PORTS"]; then
 enable_ingress_redirect_routing
 fi
}

function disable_routing() {
```

```
echo "=== Disabling routing ==="
iptables -t nat -F APPMESH_INGRESS
iptables -t nat -F APPMESH_EGRESS

if ["$APPMESH_ENABLE_IPV6" == "1"]; then
 ip6tables -t nat -F APPMESH_INGRESS
 ip6tables -t nat -F APPMESH_EGRESS
fi
}

function dump_status() {
 echo "=== iptables FORWARD table ==="
 iptables -L -v -n
 echo "=== iptables NAT table ==="
 iptables -t nat -L -v -n

 if ["$APPMESH_ENABLE_IPV6" == "1"]; then
 echo "=== ip6tables FORWARD table ==="
 ip6tables -L -v -n
 echo "=== ip6tables NAT table ==="
 ip6tables -t nat -L -v -n
 fi
}

function clean_up() {
 disable_routing
 ruleNum=$(iptables -L PREROUTING -t nat --line-numbers | grep APPMESH_INGRESS |
cut -d " " -f 1)
 iptables -t nat -D PREROUTING $ruleNum

 ruleNum=$(iptables -L OUTPUT -t nat --line-numbers | grep APPMESH_EGRESS | cut
-d " " -f 1)
 iptables -t nat -D OUTPUT $ruleNum

 iptables -t nat -X APPMESH_INGRESS
 iptables -t nat -X APPMESH_EGRESS

 if ["$APPMESH_ENABLE_IPV6" == "1"]; then
 ruleNum=$(ip6tables -L PREROUTING -t nat --line-numbers | grep
APPMESH_INGRESS | cut -d " " -f 1)
 ip6tables -t nat -D PREROUTING $ruleNum

 ruleNum=$(ip6tables -L OUTPUT -t nat --line-numbers | grep APPMESH_EGRESS |
cut -d " " -f 1)
```

```
 ip6tables -t nat -D OUTPUT $ruleNum

 ip6tables -t nat -X APPMESH_INGRESS
 ip6tables -t nat -X APPMESH_EGRESS
 fi
}

function main_loop() {
 echo "=== Entering main loop ==="
 while read -p '> ' cmd; do
 case "$cmd" in
 "quit")
 clean_up
 break
 ;;
 "status")
 dump_status
 ;;
 "enable")
 enable_routing
 ;;
 "disable")
 disable_routing
 ;;
 *)
 echo "Available commands: quit, status, enable, disable"
 ;;
 esac
 done
}

function print_config() {
 echo "=== Input configuration ==="
 env | grep APPMESH_ || true
}

print_config

initialize

if ["$APPMESH_START_ENABLED" == "1"]; then
 enable_routing
fi
```

```
main_loop
```

9. Untuk mengonfigurasi iptables aturan untuk merutekan lalu lintas aplikasi ke proxy Envoy, jalankan skrip yang Anda buat di langkah sebelumnya.

```
sudo ./envoy-networking.sh
```

10. Mulai kode aplikasi node virtual Anda.

#### Note

Untuk contoh dan penelusuran lainnya untuk App Mesh, lihat repositori contoh [App Mesh](#).

## Contoh App Mesh

#### Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk AWS App Mesh. Setelah 30 September 2026, Anda tidak lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Anda dapat menemukan end-to-end panduan yang menunjukkan AWS App Mesh contoh tindakan dan kode untuk diintegrasikan dengan berbagai AWS layanan di repositori berikut:

[Contoh App Mesh](#)

# Konsep App Mesh

## Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk AWS App Mesh. Setelah 30 September 2026, Anda tidak akan lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

App Mesh terdiri dari konsep-konsep berikut.

- [Jala Layanan](#)
- [Layanan virtual](#)
- [Gerbang virtual](#)
- [Node virtual](#)
- [Router virtual](#)

## Jala Layanan

## Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk AWS App Mesh. Setelah 30 September 2026, Anda tidak akan lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Mesh layanan adalah batas logis untuk lalu lintas jaringan antara layanan yang berada di dalamnya. Setelah membuat mesh layanan, Anda dapat membuat layanan virtual, simpul virtual, router virtual, dan rute untuk mendistribusikan lalu lintas antar aplikasi di mesh Anda.

## Membuat mesh layanan

### Note

Saat membuat Mesh, Anda harus menambahkan pemilih namespace. Jika pemilih namespace kosong, ia memilih semua ruang nama. Untuk membatasi ruang nama, gunakan label untuk mengaitkan resource App Mesh ke mesh yang dibuat.

### Konsol Manajemen AWS

Untuk membuat mesh layanan menggunakan Konsol Manajemen AWS

1. Buka konsol App Mesh di <https://console.aws.amazon.com/appmesh/>.
2. Pilih Buat mesh.
3. Untuk nama Mesh, tentukan nama untuk mesh layanan Anda.
4. (Opsional) Pilih Izinkan lalu lintas eksternal. Secara default, proxy di mesh hanya meneruskan lalu lintas antara satu sama lain. Jika Anda mengizinkan lalu lintas eksternal, proxy di mesh juga meneruskan lalu lintas TCP langsung ke layanan yang tidak digunakan dengan proxy yang ditentukan di mesh.

### Note

Jika Anda menentukan backend apa pun pada node virtual saat menggunakan `ALLOW_ALL`, Anda harus menentukan semua jalan keluar untuk node virtual itu sebagai backend. Jika tidak, `ALLOW_ALL` akan lagi berfungsi untuk node virtual itu.

5. Preferensi versi IP

Kontrol versi IP mana yang harus digunakan untuk lalu lintas di dalam mesh dengan mengaktifkan Override perilaku versi IP default. Secara default, App Mesh menggunakan berbagai versi IP.

### Note

Mesh menerapkan preferensi IP ke semua node virtual dan gateway virtual dalam mesh. Perilaku ini dapat diganti pada node virtual individual dengan mengatur

preferensi IP saat Anda membuat atau mengedit node. Preferensi IP tidak dapat diganti pada gateway virtual karena konfigurasi untuk gateway virtual yang memungkinkan mereka untuk mendengarkan keduanya IPv4 dan IPv6 lalu lintas adalah sama terlepas dari preferensi mana yang diatur pada mesh.

- Default
  - Penyelesai DNS utusan lebih suka IPv6 dan kembali ke. IPv4
  - Kami menggunakan IPv4 alamat yang dikembalikan oleh AWS Cloud Map jika tersedia dan kembali menggunakan IPv6 alamat.
  - Titik akhir yang dibuat untuk aplikasi lokal menggunakan IPv4 alamat.
  - Pendengar Utusan mengikat semua alamat. IPv4
- IPv6 lebih disukai
  - Penyelesai DNS utusan lebih suka IPv6 dan kembali ke. IPv4
  - IPv6Alamat yang dikembalikan oleh AWS Cloud Map digunakan jika tersedia dan kembali menggunakan IPv4 alamat
  - Titik akhir yang dibuat untuk aplikasi lokal menggunakan IPv6 alamat.
  - Pendengar Utusan mengikat semua dan alamat. IPv4 IPv6
- IPv4 lebih disukai
  - Penyelesai DNS utusan lebih suka IPv4 dan kembali ke. IPv6
  - Kami menggunakan IPv4 alamat yang dikembalikan oleh AWS Cloud Map jika tersedia dan kembali menggunakan IPv6 alamat.
  - Titik akhir yang dibuat untuk aplikasi lokal menggunakan IPv4 alamat.
  - Pendengar Utusan mengikat semua dan alamat. IPv4 IPv6
- IPv6 hanya
  - Penyelesai DNS Envoy hanya menggunakan. IPv6
  - Hanya IPv6 alamat yang dikembalikan oleh AWS Cloud Map yang digunakan. Jika AWS Cloud Map mengembalikan IPv4 alamat, tidak ada alamat IP yang digunakan dan hasil kosong dikembalikan ke Utusan.
  - Titik akhir yang dibuat untuk aplikasi lokal menggunakan IPv6 alamat.
  - Pendengar Utusan mengikat semua dan alamat. IPv4 IPv6

- Penyelesai DNS Envoy hanya menggunakan IPv4
  - Hanya IPv4 alamat yang dikembalikan oleh AWS Cloud Map yang digunakan. Jika AWS Cloud Map mengembalikan IPv6 alamat, tidak ada alamat IP yang digunakan dan hasil kosong dikembalikan ke Utusan.
  - Titik akhir yang dibuat untuk aplikasi lokal menggunakan IPv4 alamat.
  - Pendengar Utusan mengikat semua dan alamat. IPv4 IPv6
6. Pilih Buat mesh untuk menyelesaikan.
  7. (Opsional) Bagikan mesh dengan akun lain. Mesh bersama memungkinkan sumber daya yang dibuat oleh akun yang berbeda untuk berkomunikasi satu sama lain dalam mesh yang sama. Untuk informasi selengkapnya, lihat [Bekerja dengan jerat bersama](#).

## AWS CLI

Untuk membuat mesh menggunakan AWS CLI.

Buat mesh layanan menggunakan perintah berikut (ganti *red* nilainya dengan milik Anda sendiri):

1. 

```
aws appmesh create-mesh --mesh-name meshName
```

2. Contoh output:

```
{
 "mesh": {
 "meshName": "meshName",
 "metadata": {
 "arn": "arn:aws:appmesh:us-west-2:123456789012:mesh/meshName",
 "createdAt": "2022-04-06T08:45:50.072000-05:00",
 "lastUpdatedAt": "2022-04-06T08:45:50.072000-05:00",
 "meshOwner": "123456789012",
 "resourceOwner": "123456789012",
 "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
 "version": 1
 },
 "spec": {},
 "status": {
 "status": "ACTIVE"
 }
 }
}
```

Untuk informasi selengkapnya tentang membuat mesh dengan AWS CLI for App Mesh, lihat perintah [create-mesh](#) di referensi. AWS CLI

## Menghapus mesh

### Konsol Manajemen AWS

Untuk menghapus gateway virtual menggunakan Konsol Manajemen AWS

1. Buka konsol App Mesh di <https://console.aws.amazon.com/appmesh/>.
2. Pilih mesh yang ingin Anda hapus. Semua jerat yang Anda miliki dan yang telah [dibagikan](#) dengan Anda terdaftar.
3. Di kotak konfirmasi, ketik **delete** dan kemudian klik Hapus.

### AWS CLI

Untuk menghapus mesh menggunakan AWS CLI

1. Gunakan perintah berikut untuk menghapus mesh Anda (ganti *red* nilainya dengan milik Anda):

```
aws appmesh delete-mesh \
 --mesh-name meshName
```

2. Contoh output:

```
{
 "mesh": {
 "meshName": "meshName",
 "metadata": {
 "arn": "arn:aws:appmesh:us-west-2:123456789012:mesh/meshName",
 "createdAt": "2022-04-06T08:45:50.072000-05:00",
 "lastUpdatedAt": "2022-04-07T11:06:32.795000-05:00",
 "meshOwner": "123456789012",
 "resourceOwner": "123456789012",
 "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
 "version": 1
 },
 "spec": {},
 "status": {
```

```
 "status": "DELETED"
 }
}
```

Untuk informasi selengkapnya tentang menghapus mesh dengan App Mesh AWS CLI for, lihat perintah [delete-mesh](#) di referensi. AWS CLI

## Layanan virtual

### Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk. AWS App Mesh Setelah 30 September 2026, Anda tidak akan lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Layanan virtual adalah abstraksi dari layanan sebenarnya yang disediakan oleh simpul virtual secara langsung atau tidak langsung berdasarkan alat router virtual. Layanan dependen memanggil layanan virtual Anda berdasarkan `virtualServiceName`-nya, dan permintaan tersebut dirutekan ke simpul virtual atau router virtual yang ditentukan sebagai penyedia layanan virtual.


## Membuat layanan virtual

### Konsol Manajemen AWS

Untuk membuat layanan virtual menggunakan Konsol Manajemen AWS

1. Buka konsol App Mesh di <https://console.aws.amazon.com/appmesh/>.
2. Pilih mesh di mana Anda ingin membuat layanan virtual. Semua jerat yang Anda miliki dan yang telah [dibagikan](#) dengan Anda terdaftar.
3. Pilih Layanan virtual di navigasi kiri.
4. Pilih Buat layanan virtual.

5. Untuk nama layanan Virtual, pilih nama untuk layanan virtual Anda. Anda dapat memilih nama apa saja, tetapi nama penemuan layanan dari layanan nyata yang Anda targetkan, seperti `my-service.default.svc.cluster.local`, disarankan untuk mempermudah menghubungkan layanan virtual Anda dengan layanan nyata. Dengan cara ini Anda tidak perlu mengubah kode Anda untuk mereferensikan nama yang berbeda dari referensi kode Anda saat ini. Nama yang Anda tentukan harus diselesaikan ke alamat IP non-loopback karena wadah aplikasi harus berhasil menyelesaikan nama sebelum permintaan dikirim ke proxy Envoy. Anda dapat menggunakan alamat IP non-loopback karena aplikasi atau wadah proxy tidak berkomunikasi dengan alamat IP ini. Proxy berkomunikasi dengan layanan virtual lainnya melalui nama yang telah Anda konfigurasi untuk mereka di App Mesh, bukan melalui alamat IP yang diselesaikan nama.
6. Untuk Provider, pilih jenis penyedia untuk layanan virtual Anda:
  - Jika Anda ingin layanan virtual menyebarkan lalu lintas di beberapa node virtual, pilih Router virtual dan kemudian pilih router virtual yang akan digunakan dari menu drop-down.
  - Jika Anda ingin layanan virtual mencapai node virtual secara langsung tanpa router virtual, pilih Virtual node dan kemudian pilih node virtual yang akan digunakan dari menu drop-down.

 Note

App Mesh dapat secara otomatis membuat kebijakan coba ulang rute Utusan default untuk setiap penyedia node virtual yang Anda tentukan pada atau setelah 29 Juli 2020, meskipun Anda tidak dapat menentukan kebijakan tersebut melalui App Mesh API. Untuk informasi selengkapnya, lihat [Kebijakan coba lagi rute default](#).

- Jika Anda tidak ingin layanan virtual merutekan lalu lintas saat ini (misalnya, jika node virtual atau router virtual Anda belum ada), pilih None. Anda dapat memperbarui penyedia untuk layanan virtual ini nanti.

7. Pilih Buat layanan virtual untuk menyelesaikan.

## AWS CLI

Untuk membuat layanan virtual menggunakan AWS CLI.

Buat layanan virtual dengan penyedia node virtual menggunakan perintah berikut dan file JSON masukan (ganti *red* nilainya dengan milik Anda sendiri):

1. 

```
aws appmesh create-virtual-service \
--cli-input-json file://create-virtual-service-virtual-node.json
```

2. Isi contoh create-virtual-service-virtual -node.json:

```
{
 "meshName": "meshName",
 "spec": {
 "provider": {
 "virtualNode": {
 "virtualNodeName": "nodeName"
 }
 }
 },
 "virtualServiceName": "serviceA.svc.cluster.local"
}
```

3. Contoh output:

```
{
 "virtualService": {
 "meshName": "meshName",
 "metadata": {
 "arn": "arn:aws:appmesh:us-west-2:210987654321:mesh/meshName/virtualService/serviceA.svc.cluster.local",
 "createdAt": "2022-04-06T09:45:35.890000-05:00",
 "lastUpdatedAt": "2022-04-06T09:45:35.890000-05:00",
 "meshOwner": "123456789012",
 "resourceOwner": "210987654321",
 "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
 "version": 1
 },
 "spec": {
 "provider": {
 "virtualNode": {
 "virtualNodeName": "nodeName"
 }
 }
 },
 "status": {
```

```
 "status": "ACTIVE"
 },
 "virtualServiceName": "serviceA.svc.cluster.local"
 }
 }
```

Untuk informasi selengkapnya tentang membuat layanan virtual dengan AWS CLI for App Mesh, lihat [create-virtual-service](#) perintah di AWS CLI referensi.

## Menghapus layanan virtual

### Note

Anda tidak dapat menghapus layanan virtual yang direferensikan oleh rute gateway. Anda harus menghapus rute gateway terlebih dahulu.

### Konsol Manajemen AWS

Untuk menghapus layanan virtual menggunakan Konsol Manajemen AWS

1. Buka konsol App Mesh di <https://console.aws.amazon.com/appmesh/>.
2. Pilih mesh dari mana Anda ingin menghapus layanan virtual. Semua jerat yang Anda miliki dan yang telah [dibagikan](#) dengan Anda terdaftar.
3. Pilih Layanan virtual di navigasi kiri.
4. Pilih layanan virtual yang ingin Anda hapus dan klik Hapus di sudut kanan atas. Anda hanya dapat menghapus gateway virtual di mana akun Anda terdaftar sebagai pemilik Sumber Daya.
5. Di kotak konfirmasi, ketik **delete** dan kemudian klik Hapus.

### AWS CLI

Untuk menghapus layanan virtual menggunakan AWS CLI

1. Gunakan perintah berikut untuk menghapus layanan virtual Anda (ganti *red* nilainya dengan milik Anda sendiri):

```
aws appmesh delete-virtual-service \
 --mesh-name meshName \
 --virtual-service-name serviceA.svc.cluster.local
```

## 2. Contoh output:

```
{
 "virtualService": {
 "meshName": "meshName",
 "metadata": {
 "arn": "arn:aws:appmesh:us-west-2:210987654321:mesh/meshName/
virtualService/serviceA.svc.cluster.local",
 "createdAt": "2022-04-06T09:45:35.890000-05:00",
 "lastUpdatedAt": "2022-04-07T10:39:42.772000-05:00",
 "meshOwner": "123456789012",
 "resourceOwner": "210987654321",
 "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
 "version": 2
 },
 "spec": {
 "provider": {
 "virtualNode": {
 "virtualNodeName": "nodeName"
 }
 }
 },
 "status": {
 "status": "DELETED"
 },
 "virtualServiceName": "serviceA.svc.cluster.local"
 }
}
```

Untuk informasi selengkapnya tentang menghapus layanan virtual dengan AWS CLI for App Mesh, lihat [delete-virtual-service](#) perintah di AWS CLI referensi.

## Gerbang virtual

### Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk AWS App Mesh. Setelah 30 September 2026, Anda tidak akan lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Gateway virtual memungkinkan sumber daya yang berada di luar mesh Anda untuk berkomunikasi dengan sumber daya yang ada di dalam mesh Anda. Gateway virtual mewakili proxy Envoy yang berjalan di layanan Amazon ECS, di layanan Kubernetes, atau di instans Amazon EC2. Tidak seperti node virtual, yang mewakili Utusan yang berjalan dengan aplikasi, gateway virtual mewakili Utusan yang digunakan dengan sendirinya.

Sumber daya eksternal harus dapat menyelesaikan nama DNS ke alamat IP yang ditetapkan ke layanan atau instance yang menjalankan Envoy. Envoy kemudian dapat mengakses semua konfigurasi App Mesh untuk sumber daya yang ada di dalam mesh. Konfigurasi untuk menangani permintaan masuk di Virtual Gateway ditentukan menggunakan [Rute Gateway](#).

### Important

Gateway virtual dengan HTTP atau HTTP2 listener menulis ulang nama host permintaan masuk ke nama Layanan Virtual target Gateway Route, dan awalan yang cocok dari Rute Gateway ditulis ulang, secara default. / Misalnya, jika Anda telah mengonfigurasi awalan pencocokan rute Gateway ke `/chapter`, dan, jika permintaan masuk `/chapter/1`, permintaan akan ditulis ulang. /1 Untuk mengonfigurasi penulisan ulang, lihat bagian [Membuat rute gateway](#) dari Rute Gateway.

Saat membuat gateway virtual, `proxyConfiguration` dan `user` harus dikonfigurasi.

Untuk menyelesaikan end-to-end penelusuran, lihat [Mengonfigurasi Gateway Masuk](#).

## Membuat gateway virtual

### Note


Saat membuat Gateway Virtual, Anda harus menambahkan pemilih namespace dengan label untuk mengidentifikasi daftar ruang nama yang dapat digunakan untuk mengaitkan Rute Gateway ke Gateway Virtual yang dibuat.

### Konsol Manajemen AWS

Untuk membuat gateway virtual menggunakan Konsol Manajemen AWS

1. Buka konsol App Mesh di <https://console.aws.amazon.com/appmesh/>.
2. Pilih mesh di mana Anda ingin membuat gateway virtual. Semua jerat yang Anda miliki dan yang telah [dibagikan](#) dengan Anda terdaftar.
3. Pilih gateway Virtual di navigasi kiri.
4. Pilih Buat gateway virtual.
5. Untuk nama gateway Virtual, masukkan nama untuk gateway virtual Anda.
6. (Opsional, tetapi disarankan) Konfigurasi default kebijakan Klien.
  - a. (Opsional) Pilih Terapkan TLS jika Anda ingin gateway hanya berkomunikasi dengan layanan virtual menggunakan Transport Layer Security (TLS).
  - b. (Opsional) Untuk Port, tentukan satu atau beberapa port tempat Anda ingin menerapkan komunikasi TLS dengan layanan virtual.
  - c. Untuk metode Validasi, pilih salah satu opsi berikut. Sertifikat yang Anda tentukan harus sudah ada dan memenuhi persyaratan khusus. Untuk informasi selengkapnya, lihat [Persyaratan sertifikat](#).
    - AWS Private Certificate Authorityhosting — Pilih satu atau lebih Sertifikat yang ada.
    - Hosting Envoy Secret Discovery Service (SDS) - Masukkan nama rahasia yang diambil Utusan menggunakan Secret Discovery Service.
    - Hosting file lokal - Tentukan jalur ke file rantai Sertifikat pada sistem file tempat Utusan digunakan.
  - d. (Opsional) Masukkan Nama Alternatif Subjek. Untuk menambahkan tambahan SANs, pilih Tambahkan SAN. SANs harus diformat FQDN atau URI.

- e. (Opsional) Pilih Berikan sertifikat klien dan salah satu opsi di bawah ini untuk memberikan sertifikat klien saat server memintanya dan mengaktifkan otentikasi TLS bersama. Untuk mempelajari selengkapnya tentang TLS timbal balik, lihat dokumen [Otentikasi TLS Mutual](#) App Mesh.
  - Hosting Envoy Secret Discovery Service (SDS) - Masukkan nama rahasia yang diambil Utusan menggunakan Secret Discovery Service.
  - Hosting file lokal - Tentukan jalur ke file rantai Sertifikat, serta kunci Pribadi, pada sistem file tempat Utusan digunakan. Untuk mengetahui selengkapnya tentang penerapan mesh dengan aplikasi sampel menggunakan enkripsi dengan file lokal, lihat [Mengonfigurasi TLS dengan Sertifikat TLS yang Disediakan File](#) pada. end-to-end GitHub
7. (Opsional) Untuk mengkonfigurasi logging, pilih Logging. Masukkan jalur log akses HTTP yang ingin digunakan oleh Envoy. Kami merekomendasikan `/dev/stdout` jalur sehingga Anda dapat menggunakan driver log Docker untuk mengeksport log Utusan Anda ke layanan seperti Amazon Logs. CloudWatch

 Note

Log masih harus dicerna oleh agen dalam aplikasi Anda dan dikirim ke tujuan. Jalur file ini hanya menginstruksikan Utusan ke mana harus mengirim log.

8. Konfigurasi Listener.
  - a. Pilih Protokol dan tentukan Port tempat Utusan mendengarkan lalu lintas. Pendengar http mengizinkan transisi koneksi ke soket web. Anda dapat mengklik Add Listener untuk menambahkan beberapa pendengar. Tombol Hapus akan menghapus pendengar itu.
  - b. (Opsional) Aktifkan kumpulan koneksi

Penggabungan koneksi membatasi jumlah koneksi yang dapat dibuat oleh Utusan Gateway Virtual secara bersamaan. Hal ini dimaksudkan untuk melindungi instans Utusan Anda dari kewalahan dengan koneksi dan memungkinkan Anda menyesuaikan pembentukan lalu lintas untuk kebutuhan aplikasi Anda.

Anda dapat mengonfigurasi pengaturan kumpulan koneksi sisi tujuan untuk pendengar gateway virtual. App Mesh menetapkan pengaturan kumpulan koneksi sisi klien ke tak terbatas secara default, menyederhanakan konfigurasi mesh.

**Note**

Protokol `connectionPool` dan `connectionPool` `PortMapping` harus sama. Jika protokol pendengar Anda adalah `grpc` atau `http2`, tentukan `maxRequests` saja. Jika protokol pendengar Anda `http`, Anda dapat menentukan keduanya `maxConnections` dan `maxPendingRequests`.

- Untuk Koneksi maksimum, tentukan jumlah maksimum koneksi keluar.
  - Untuk permintaan Maksimum, tentukan jumlah maksimum permintaan paralel yang dapat dibuat dengan Virtual Gateway Envoy.
  - (Opsional) Untuk permintaan tertunda maksimum, tentukan jumlah permintaan yang meluap setelah Koneksi maksimum yang antrian oleh Utusan. Nilai default-nya adalah 2147483647.
- c. (Opsional) Jika Anda ingin mengonfigurasi pemeriksaan kesehatan untuk pendengar Anda, pilih Aktifkan pemeriksaan kesehatan.

Kebijakan pemeriksaan kesehatan bersifat opsional, tetapi jika Anda menentukan nilai apa pun untuk kebijakan kesehatan, maka Anda harus menentukan nilai untuk ambang batas Sehat, Interval pemeriksaan Kesehatan, Protokol pemeriksaan Kesehatan, Periode waktu tunggu, dan ambang tidak sehat.

- Untuk protokol pemeriksaan Kesehatan, pilih protokol. Jika Anda memilih `grpc`, maka layanan Anda harus sesuai dengan Protokol Pemeriksaan [Kesehatan GRPC](#).
- Untuk port pemeriksaan Kesehatan, tentukan port tempat pemeriksaan kesehatan harus dijalankan.
- Untuk ambang sehat, tentukan jumlah pemeriksaan kesehatan yang berhasil berturut-turut yang harus dilakukan sebelum menyatakan pendengar sehat.
- Untuk interval pemeriksaan Kesehatan, tentukan periode waktu dalam milidetik antara setiap eksekusi pemeriksaan kesehatan.
- Untuk Path, tentukan jalur tujuan untuk permintaan pemeriksaan kesehatan. Nilai ini hanya digunakan jika protokol pemeriksaan Kesehatan adalah `http` atau `http2`. Nilai diabaikan untuk protokol lain.
- Untuk periode Timeout, tentukan jumlah waktu untuk menunggu saat menerima respons dari pemeriksaan kesehatan dalam milidetik.

- Untuk ambang tidak sehat, tentukan jumlah pemeriksaan kesehatan gagal berturut-turut yang harus dilakukan sebelum menyatakan pendengar tidak sehat.
- d. (Opsional) Jika Anda ingin menentukan apakah klien berkomunikasi dengan gateway virtual ini menggunakan TLS, lalu pilih Aktifkan penghentian TLS.
- Untuk Mode, pilih mode yang Anda inginkan untuk dikonfigurasi TLS pada pendengar.
  - Untuk metode Sertifikat, pilih salah satu opsi berikut. Sertifikat harus memenuhi persyaratan khusus. Untuk informasi selengkapnya, lihat [Persyaratan sertifikat](#).
    - AWS Certificate Manager hosting — Pilih Sertifikat yang ada.
    - Hosting Envoy Secret Discovery Service (SDS) - Masukkan nama rahasia yang diambil Utusan menggunakan Secret Discovery Service.
    - Hosting file lokal - Tentukan jalur ke rantai Sertifikat dan file kunci pribadi pada sistem file tempat Utusan digunakan.
  - (Opsional) Pilih Memerlukan sertifikat klien dan salah satu opsi di bawah ini untuk mengaktifkan otentikasi TLS timbal balik jika klien memberikan sertifikat. Untuk mempelajari selengkapnya tentang TLS timbal balik, lihat dokumen [Otentikasi TLS Mutual](#) App Mesh.
    - Hosting Envoy Secret Discovery Service (SDS) - Masukkan nama rahasia yang diambil Utusan menggunakan Secret Discovery Service.
    - Hosting file lokal - Tentukan jalur ke file rantai Sertifikat pada sistem file tempat Utusan digunakan.
  - (Opsional) Masukkan Nama Alternatif Subjek. Untuk menambahkan tambahan SANs, pilih Tambahkan SAN. SANs harus diformat FQDN atau URI.
9. Pilih Buat gateway virtual untuk menyelesaikan.

## AWS CLI

Untuk membuat gateway virtual menggunakan file AWS CLI.

Buat gateway virtual menggunakan perintah berikut dan masukan JSON (ganti *red* nilai dengan milik Anda sendiri):

1. 

```
aws appmesh create-virtual-gateway \
--mesh-name meshName \
--virtual-gateway-name virtualGatewayName \

```

```
--cli-input-json file://create-virtual-gateway.json
```

## 2. Isi contoh create-virtual-gateway .json:

```
{
 "spec": {
 "listeners": [
 {
 "portMapping": {
 "port": 9080,
 "protocol": "http"
 }
 }
]
 }
}
```

## 3. Contoh output:

```
{
 "virtualGateway": {
 "meshName": "meshName",
 "metadata": {
 "arn": "arn:aws:appmesh:us-west-2:123456789012:mesh/meshName/virtualGateway/virtualGatewayName",
 "createdAt": "2022-04-06T10:42:42.015000-05:00",
 "lastUpdatedAt": "2022-04-06T10:42:42.015000-05:00",
 "meshOwner": "123456789012",
 "resourceOwner": "123456789012",
 "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
 "version": 1
 },
 "spec": {
 "listeners": [
 {
 "portMapping": {
 "port": 9080,
 "protocol": "http"
 }
 }
]
 },
 "status": {
```

```
 "status": "ACTIVE"
 },
 "virtualGatewayName": "virtualGatewayName"
 }
 }
```

Untuk informasi selengkapnya tentang membuat gateway virtual dengan AWS CLI for App Mesh, lihat [create-virtual-gateway](#) perintah di AWS CLI referensi.

## Menyebarkan gateway virtual

[Menerapkan layanan Amazon ECS atau Kubernetes yang hanya berisi container Envoy.](#) Anda juga dapat menerapkan container Envoy di instans Amazon. EC2 Untuk informasi selengkapnya, lihat [Memulai App Mesh dan Amazon EC2](#). Untuk informasi selengkapnya tentang cara menerapkan di Amazon ECS, lihat [Memulai App Mesh dan Amazon ECS atau Memulai App Mesh AWS dan Kubernetes untuk diterapkan ke Kubernetes](#). Anda perlu mengatur variabel `APPESH_RESOURCE_ARN` lingkungan ke `mesh/mesh-name/virtualGateway/virtual-gateway-name` dan Anda tidak boleh menentukan konfigurasi proxy sehingga lalu lintas proxy tidak diarahkan ke dirinya sendiri. Secara default, App Mesh menggunakan nama sumber daya yang Anda tentukan `APPESH_RESOURCE_ARN` saat Envoy merujuk dirinya sendiri dalam metrik dan jejak. Anda dapat menimpa perilaku ini dengan mengatur variabel lingkungan `APPESH_RESOURCE_CLUSTER` dengan nama Anda sendiri.

Kami menyarankan Anda menerapkan beberapa instance kontainer dan menyiapkan Network Load Balancer untuk memuat lalu lintas keseimbangan ke instans. Nama penemuan layanan penyeimbang beban adalah nama yang Anda inginkan untuk digunakan layanan eksternal untuk mengakses sumber daya yang ada di mesh, seperti `myapp.example.com`. Untuk informasi selengkapnya lihat [Membuat Network Load Balancer](#) (Amazon ECS), [Membuat Load Balancer Eksternal](#) (Kubernetes), atau [Tutorial: Meningkatkan ketersediaan aplikasi Anda di Amazon. EC2](#) Anda juga dapat menemukan lebih banyak contoh dan penelusuran di contoh [App](#) Mesh kami.

Aktifkan otorisasi proxy untuk Utusan. Untuk informasi selengkapnya, lihat [Otorisasi Proxy Utusan](#).

# Menghapus gateway virtual

## Konsol Manajemen AWS

Untuk menghapus gateway virtual menggunakan Konsol Manajemen AWS

1. Buka konsol App Mesh di <https://console.aws.amazon.com/appmesh/>.
2. Pilih mesh dari mana Anda ingin menghapus gateway virtual. Semua jerat yang Anda miliki dan yang telah [dibagikan](#) dengan Anda terdaftar.
3. Pilih gateway Virtual di navigasi kiri.
4. Pilih gateway virtual yang ingin Anda hapus dan pilih Hapus. Anda tidak dapat menghapus gateway virtual jika memiliki rute gateway terkait. Anda harus menghapus rute gateway terkait terlebih dahulu. Anda hanya dapat menghapus gateway virtual di mana akun Anda terdaftar sebagai pemilik Sumber Daya.
5. Di kotak konfirmasi, ketik **delete** lalu pilih Hapus.

## AWS CLI

Untuk menghapus gateway virtual menggunakan AWS CLI

1. Gunakan perintah berikut untuk menghapus gateway virtual Anda (ganti *red* nilainya dengan milik Anda sendiri):

```
aws appmesh delete-virtual-gateway \
 --mesh-name meshName \
 --virtual-gateway-name virtualGatewayName
```

2. Contoh output:

```
{
 "virtualGateway": {
 "meshName": "meshName",
 "metadata": {
 "arn": "arn:aws:appmesh:us-west-2:123456789012:mesh/meshName/
virtualGateway/virtualGatewayName",
 "createdAt": "2022-04-06T10:42:42.015000-05:00",
 "lastUpdatedAt": "2022-04-07T10:57:22.638000-05:00",
 "meshOwner": "123456789012",
 "resourceOwner": "123456789012",
```

```
 "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
 "version": 2
 },
 "spec": {
 "listeners": [
 {
 "portMapping": {
 "port": 9080,
 "protocol": "http"
 }
 }
]
 },
 "status": {
 "status": "DELETED"
 },
 "virtualGatewayName": "virtualGatewayName"
}
}
```

Untuk informasi selengkapnya tentang menghapus gateway virtual dengan AWS CLI for App Mesh, lihat [delete-virtual-gateway](#) perintah di AWS CLI referensi.

## Rute gerbang

### Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk AWS App Mesh. Setelah 30 September 2026, Anda tidak lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Rute gateway dilampirkan ke gateway virtual dan merutekan lalu lintas ke layanan virtual yang ada. Jika rute cocok dengan permintaan, ia dapat mendistribusikan lalu lintas ke layanan virtual target. Topik ini membantu Anda bekerja dengan rute gateway di mesh layanan.

## Membuat rute gateway

### Konsol Manajemen AWS

Untuk membuat rute gateway menggunakan Konsol Manajemen AWS

1. Buka konsol App Mesh di <https://console.aws.amazon.com/appmesh/>.
2. Pilih mesh di mana Anda ingin membuat rute gateway. Semua jerat yang Anda miliki dan yang telah [dibagikan](#) dengan Anda terdaftar.
3. Pilih gateway Virtual di navigasi kiri.
4. Pilih gateway virtual yang ingin Anda kaitkan dengan rute gateway baru. Jika tidak ada yang terdaftar, maka Anda perlu [membuat gateway virtual](#) terlebih dahulu. Anda hanya dapat membuat rute gateway untuk gateway virtual yang akun Anda terdaftar sebagai pemilik Sumber Daya.
5. Dalam tabel rute Gateway, pilih Buat rute gateway.
6. Untuk nama rute Gateway, tentukan nama yang akan digunakan untuk rute gateway Anda.
7. Untuk jenis rute Gateway pilih http, http2, atau grpc.
8. Pilih nama layanan Virtual yang ada. Jika tidak ada yang terdaftar, maka Anda perlu membuat [layanan virtual](#) terlebih dahulu.
9. Pilih port yang sesuai dengan target untuk port penyedia layanan Virtual. Port penyedia layanan virtual diperlukan ketika penyedia (router atau node) dari layanan virtual yang dipilih memiliki banyak pendengar.
10. (Opsional) Untuk Prioritas, tentukan prioritas untuk rute gateway ini.
11. Untuk konfigurasi Match, tentukan:
  - Jika http/http2 adalah jenis yang dipilih:
    - (Opsional) Metode - Menentukan header metode yang akan dicocokkan dalam permintaan http/ http2 masuk.
    - (Opsional) Pencocokan port - Cocokkan port untuk lalu lintas masuk. Pencocokan port diperlukan jika gateway virtual ini memiliki banyak pendengar.
    - (Opsional) Nama host Exact/Suffix - Menentukan nama host yang harus dicocokkan pada permintaan masuk untuk rute ke layanan virtual target.
    - (Opsional) Prefix/Exact/Regexpjalur - Metode pencocokan jalur URL.
      - Pencocokan awalan - Permintaan yang cocok oleh rute gateway ditulis ulang ke nama layanan virtual target dan awalan yang cocok ditulis ulang, secara default.

/ Bergantung pada bagaimana Anda mengkonfigurasi layanan virtual Anda, itu bisa menggunakan router virtual untuk merutekan permintaan ke node virtual yang berbeda, berdasarkan awalan atau header tertentu.

#### Important

- Anda tidak dapat menentukan salah satu `/aws-appmesh*` atau `/aws-app-mesh*` untuk kecocokan Awalan. Awalan ini dicadangkan untuk penggunaan internal App Mesh di masa mendatang.
- Jika beberapa rute gateway ditentukan, maka permintaan dicocokkan dengan rute dengan awalan terpanjang. Misalnya, jika ada dua rute gateway, dengan satu memiliki awalan `/chapter` dan satu memiliki awalan `/`, maka permintaan untuk `www.example.com/chapter/` akan dicocokkan dengan rute gateway dengan awalan `/chapter`

#### Note

Jika Anda mengaktifkan pencocokan berbasis Path/ Prefix, App Mesh mengaktifkan normalisasi jalur ([normalize\\_path](#) dan [merge\\_slashes](#)) untuk meminimalkan kemungkinan kerentanan kebingungan jalur.

Kerentanan kebingungan jalur terjadi ketika pihak yang berpartisipasi dalam permintaan menggunakan representasi jalur yang berbeda.

- Pencocokan tepat - Parameter yang tepat menonaktifkan pencocokan sebagian untuk rute dan memastikan bahwa itu hanya mengembalikan rute jika jalurnya sama dengan URL saat ini.
- Pencocokan Regex - Digunakan untuk menggambarkan pola di mana beberapa URLs mungkin benar-benar mengidentifikasi satu halaman di situs web.
- (Opsional) Parameter kueri - Bidang ini memungkinkan Anda untuk mencocokkan pada parameter kueri.
- (Opsional) Header - Menentukan header untuk http dan http2. Itu harus sesuai dengan permintaan yang masuk untuk merutekan ke layanan virtual target..
- Jika `grpc` adalah tipe yang dipilih:

- Jenis pencocokan nama host dan (opsional) Pencocokan Exact/Suffix - Menentukan nama host yang harus dicocokkan pada permintaan masuk untuk merutekan ke layanan virtual target.
- nama layanan grpc - Layanan grpc bertindak sebagai API untuk aplikasi Anda dan didefinisikan dengan. ProtoBuf

 Important

Anda tidak dapat menentukan `/aws.app-mesh*` atau `aws.appmesh` untuk nama Layanan. Nama layanan ini dicadangkan untuk penggunaan internal App Mesh di masa mendatang.

- (Opsional) Metadata - Menentukan metadata untuk grpc. Ini harus sesuai dengan permintaan yang masuk untuk merutekan ke layanan virtual target.

## 12. (Opsional) Untuk konfigurasi Tulis Ulang:

- Jika http/http2 adalah jenis yang dipilih:
  - Jika Awalan adalah jenis kecocokan yang dipilih:
    - Ganti penulisan ulang otomatis nama host - Secara default nama host ditulis ulang ke nama layanan virtual target.
    - Ganti penulisan ulang otomatis awalan - Saat diaktifkan, Prefix rewrite menentukan nilai awalan yang ditulis ulang.
  - Jika Exact Path adalah jenis kecocokan yang dipilih:
    - Ganti penulisan ulang otomatis nama host - secara default nama host ditulis ulang ke nama layanan virtual target.
    - Path rewrite - Menentukan nilai jalur ditulis ulang. Tidak ada jalur default.
  - Jika Regex Path adalah jenis kecocokan yang dipilih:
    - Ganti penulisan ulang otomatis nama host - secara default nama host ditulis ulang ke nama layanan virtual target.
    - Path rewrite - Menentukan nilai jalur ditulis ulang. Tidak ada jalur default.
- Jika grpc adalah tipe yang dipilih:
  - Ganti penulisan ulang otomatis nama host - Secara default nama host ditulis ulang ke nama layanan virtual target.

## 13. Pilih Buat rute gateway untuk menyelesaikan.

## AWS CLI

Untuk membuat rute gateway menggunakan file AWS CLI.

Buat rute gateway menggunakan perintah berikut dan masukan JSON (ganti *red* nilainya dengan milik Anda sendiri):

```
1. aws appmesh create-virtual-gateway \
 --mesh-name meshName \
 --virtual-gateway-name virtualGatewayName \
 --gateway-route-name gatewayRouteName \
 --cli-input-json file://create-gateway-route.json
```

2. Isi contoh create-gateway-route .json:

```
{
 "spec": {
 "httpRoute" : {
 "match" : {
 "prefix" : "/"
 },
 "action" : {
 "target" : {
 "virtualService": {
 "virtualServiceName": "serviceA.svc.cluster.local"
 }
 }
 }
 }
 }
}
```

3. Contoh output:

```
{
 "gatewayRoute": {
 "gatewayRouteName": "gatewayRouteName",
 "meshName": "meshName",
 "metadata": {
 "arn": "arn:aws:appmesh:us-west-2:210987654321:mesh/meshName/
virtualGateway/virtualGatewayName/gatewayRoute/gatewayRouteName",
 "createdAt": "2022-04-06T11:05:32.100000-05:00",
 "lastUpdatedAt": "2022-04-06T11:05:32.100000-05:00",
 }
 }
}
```

```
 "meshOwner": "123456789012",
 "resourceOwner": "210987654321",
 "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
 "version": 1
 },
 "spec": {
 "httpRoute": {
 "action": {
 "target": {
 "virtualService": {
 "virtualServiceName": "serviceA.svc.cluster.local"
 }
 }
 },
 "match": {
 "prefix": "/"
 }
 }
 },
 "status": {
 "status": "ACTIVE"
 },
 "virtualGatewayName": "gatewayName"
}
}
```

Untuk informasi selengkapnya tentang membuat rute gateway dengan App Mesh AWS CLI for, lihat [create-gateway-route](#) perintah di AWS CLI referensi.

## Menghapus rute gateway

### Konsol Manajemen AWS

Untuk menghapus rute gateway menggunakan Konsol Manajemen AWS

1. Buka konsol App Mesh di <https://console.aws.amazon.com/appmesh/>.
2. Pilih mesh dari mana Anda ingin menghapus rute gateway. Semua jerat yang Anda miliki dan yang telah [dibagikan](#) dengan Anda terdaftar.
3. Pilih gateway Virtual di navigasi kiri.
4. Pilih gateway virtual dari mana Anda ingin menghapus rute gateway.

5. Dalam tabel rute Gateway, pilih rute gateway yang ingin Anda hapus dan pilih Hapus. Anda hanya dapat menghapus rute gateway jika akun Anda terdaftar sebagai pemilik Sumber Daya.
6. Di kotak konfirmasi, ketik **delete** dan kemudian klik Hapus.

## AWS CLI

Untuk menghapus rute gateway menggunakan AWS CLI

1. Gunakan perintah berikut untuk menghapus rute gateway Anda (ganti *red* nilainya dengan milik Anda sendiri):

```
aws appmesh delete-gateway-route \
 --mesh-name meshName \
 --virtual-gateway-name virtualGatewayName \
 --gateway-route-name gatewayRouteName
```

2. Contoh output:

```
{
 "gatewayRoute": {
 "gatewayRouteName": "gatewayRouteName",
 "meshName": "meshName",
 "metadata": {
 "arn": "arn:aws:appmesh:us-west-2:210987654321:mesh/meshName/
virtualGateway/virtualGatewayName/gatewayRoute/gatewayRouteName",
 "createdAt": "2022-04-06T11:05:32.100000-05:00",
 "lastUpdatedAt": "2022-04-07T10:36:33.191000-05:00",
 "meshOwner": "123456789012",
 "resourceOwner": "210987654321",
 "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
 "version": 2
 },
 "spec": {
 "httpRoute": {
 "action": {
 "target": {
 "virtualService": {
 "virtualServiceName": "serviceA.svc.cluster.local"
 }
 }
 }
 }
 }
 }
}
```

```
 },
 "match": {
 "prefix": "/"
 }
 },
 "status": {
 "status": "DELETED"
 },
 "virtualGatewayName": "virtualGatewayName"
}
}
```

Untuk informasi selengkapnya tentang menghapus rute gateway dengan App Mesh AWS CLI for, lihat [delete-gateway-route](#) perintah di AWS CLI referensi.

## Node virtual

### Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk AWS App Mesh. Setelah 30 September 2026, Anda tidak lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Node virtual bertindak sebagai pointer logis ke grup tugas tertentu, seperti layanan Amazon ECS atau penerapan Kubernetes. Saat Anda membuat node virtual, Anda harus menentukan metode penemuan layanan untuk grup tugas Anda. Setiap lalu lintas masuk yang diharapkan oleh node virtual Anda ditentukan sebagai pendengar. Setiap layanan virtual yang node virtual mengirimkan lalu lintas keluar ke ditentukan sebagai backend.

Metadata respons untuk node virtual baru Anda berisi Amazon Resource Name (ARN) yang terkait dengan node virtual. Tetapkan nilai ini sebagai variabel `APPMESH_RESOURCE_ARN` lingkungan untuk container proxy Envoy grup tugas Anda dalam definisi tugas Amazon ECS atau spesifikasi pod Kubernetes. Misalnya, nilainya bisa `arn:aws:appmesh:us-west-2:111122223333:mesh/myMesh/virtualNode/myVirtualNode`. Ini kemudian

dipetakan ke parameter Envoy node `.id` dan `node.cluster`. Anda harus menggunakan `1.15.0` atau lebih baru dari gambar Envoy saat mengatur variabel ini. Untuk informasi selengkapnya tentang variabel App Mesh Envoy, lihat [Utusan](#)

#### Note

Secara default, App Mesh menggunakan nama sumber daya yang Anda tentukan `APPMESH_RESOURCE_ARN` saat Envoy merujuk dirinya sendiri dalam metrik dan jejak. Anda dapat menimpa perilaku ini dengan mengatur variabel lingkungan `APPMESH_RESOURCE_CLUSTER` dengan nama Anda sendiri.

## Membuat simpul virtual

### Konsol Manajemen AWS

Untuk membuat node virtual menggunakan Konsol Manajemen AWS

1. Buka konsol App Mesh di <https://console.aws.amazon.com/appmesh/>.
2. Pilih mesh tempat Anda ingin membuat simpul virtual. Semua jerat yang Anda miliki dan yang telah [dibagikan](#) dengan Anda terdaftar.
3. Pilih Virtual node di navigasi kiri.
4. Pilih Buat simpul virtual dan kemudian tentukan pengaturan untuk node virtual Anda.
5. Untuk nama node Virtual, masukkan nama untuk node virtual Anda.
6. Untuk metode Penemuan layanan, pilih salah satu opsi berikut:
  - DNS — Tentukan nama host DNS dari layanan aktual yang diwakili oleh node virtual. Proxy utusan digunakan di VPC Amazon. Proxy mengirimkan permintaan resolusi nama ke server DNS yang dikonfigurasi untuk VPC. Jika nama host diselesaikan, server DNS mengembalikan satu atau lebih alamat IP. Untuk informasi selengkapnya tentang setelan DNS VPC, lihat [Menggunakan DNS dengan](#) VPC Anda. Untuk tipe respons DNS (opsional), tentukan jenis titik akhir yang dikembalikan oleh penyelesai DNS. Load Balancer berarti bahwa resolver DNS mengembalikan set titik akhir loadbalanced. Endpoints berarti bahwa DNS resolver mengembalikan semua endpoint. Secara default, tipe respons diasumsikan sebagai Load Balancer.

**Note**

Jika Anda menggunakan Route53, Anda harus menggunakan Load Balancer.

- AWS Cloud Map— Tentukan nama Layanan yang ada dan HTTP Namespace. Secara opsional, Anda juga dapat menentukan atribut yang dapat dikueri AWS Cloud Map App Mesh dengan memilih Tambah baris dan menentukan Kunci dan Nilai. Hanya contoh yang cocok dengan semua key/value pasangan yang ditentukan akan dikembalikan. Untuk menggunakannya AWS Cloud Map, akun Anda harus memiliki peran `AWSServiceRoleForAppMesh` [terkait layanan](#). Untuk informasi selengkapnya AWS Cloud Map, lihat [Panduan AWS Cloud Map Pengembang](#).
- Tidak ada - Pilih jika node virtual Anda tidak mengharapakan lalu lintas masuk.

## 7. Preferensi versi IP


Kontrol versi IP mana yang harus digunakan untuk lalu lintas di dalam mesh dengan mengaktifkan Override perilaku versi IP default. Secara default, App Mesh menggunakan berbagai versi IP.

**Note**

Mengatur preferensi IP pada node virtual hanya mengesampingkan preferensi IP yang ditetapkan untuk mesh pada node spesifik ini.

- Default
  - Penyelesai DNS utusan lebih suka IPv6 dan kembali ke. IPv4
  - Kami menggunakan IPv4 alamat yang dikembalikan oleh AWS Cloud Map jika tersedia dan kembali menggunakan IPv6 alamat.
  - Titik akhir yang dibuat untuk aplikasi lokal menggunakan IPv4 alamat.
  - Pendengar Utusan mengikat semua alamat. IPv4
- IPv6 lebih disukai
  - Penyelesai DNS utusan lebih suka IPv6 dan kembali ke. IPv4
  - IPv6Alamat yang dikembalikan oleh AWS Cloud Map digunakan jika tersedia dan kembali menggunakan IPv4 alamat
  - Titik akhir yang dibuat untuk aplikasi lokal menggunakan IPv6 alamat.

- Pendengar Utusan mengikat semua dan alamat. IPv4 IPv6
  - IPv4 lebih disukai
    - Penyelesai DNS utusan lebih suka IPv4 dan kembali ke. IPv6
    - Kami menggunakan IPv4 alamat yang dikembalikan oleh AWS Cloud Map jika tersedia dan kembali menggunakan IPv6 alamat.
    - Titik akhir yang dibuat untuk aplikasi lokal menggunakan IPv4 alamat.
    - Pendengar Utusan mengikat semua dan alamat. IPv4 IPv6
  - IPv6 hanya
    - Penyelesai DNS Envoy hanya menggunakan. IPv6
    - Hanya IPv6 alamat yang dikembalikan oleh AWS Cloud Map yang digunakan. Jika AWS Cloud Map mengembalikan IPv4 alamat, tidak ada alamat IP yang digunakan dan hasil kosong dikembalikan ke Utusan.
    - Titik akhir yang dibuat untuk aplikasi lokal menggunakan IPv6 alamat.
    - Pendengar Utusan mengikat semua dan alamat. IPv4 IPv6
  - IPv4 hanya
    - Penyelesai DNS Envoy hanya menggunakan. IPv4
    - Hanya IPv4 alamat yang dikembalikan oleh AWS Cloud Map yang digunakan. Jika AWS Cloud Map mengembalikan IPv6 alamat, tidak ada alamat IP yang digunakan dan hasil kosong dikembalikan ke Utusan.
    - Titik akhir yang dibuat untuk aplikasi lokal menggunakan IPv4 alamat.
    - Pendengar Utusan mengikat semua dan alamat. IPv4 IPv6
8. (Opsional) Default kebijakan klien — Konfigurasi persyaratan default saat berkomunikasi ke layanan virtual backend.

 Note

- Jika Anda ingin mengaktifkan Transport Layer Security (TLS) untuk node virtual yang ada, maka kami sarankan Anda membuat node virtual baru, yang mewakili layanan yang sama dengan node virtual yang ada, untuk mengaktifkan TLS. Kemudian secara bertahap mengalihkan lalu lintas ke node virtual baru menggunakan router virtual dan rute. Untuk informasi selengkapnya tentang membuat rute dan menyesuaikan bobot untuk transisi, lihat [Rute](#). Jika Anda

kemungkinan proxy Envoy klien hilir akan menerima konteks validasi TLS sebelum proxy Envoy untuk node virtual yang telah Anda perbarui menerima sertifikat. Hal ini dapat menyebabkan kesalahan negosiasi TLS pada proxy Utusan hilir.

- [Otorisasi proxy](#) harus diaktifkan untuk proxy Envoy yang digunakan dengan aplikasi yang diwakili oleh node virtual layanan backend. Kami menyarankan bahwa ketika Anda mengaktifkan otorisasi proxy, Anda membatasi akses hanya ke node virtual yang node virtual ini berkomunikasi dengan.

- (Opsional) Pilih Terapkan TLS jika Anda ingin meminta node virtual untuk berkomunikasi dengan semua backend menggunakan Transport Layer Security (TLS).
- (Opsional) Jika Anda hanya ingin meminta penggunaan TLS untuk satu atau lebih port tertentu, maka masukkan nomor di Port. Untuk menambahkan port tambahan, pilih Tambah port. Jika Anda tidak menentukan port apa pun, TLS diberlakukan untuk semua port.
- Untuk metode Validasi, pilih salah satu opsi berikut. Sertifikat yang Anda tentukan harus sudah ada dan memenuhi persyaratan khusus. Untuk informasi selengkapnya, lihat [Persyaratan sertifikat](#).
  - AWS Private Certificate Authorityhosting — Pilih satu atau lebih Sertifikat yang ada. Untuk selengkapnya tentang penerapan mesh dengan aplikasi sampel menggunakan enkripsi dengan sertifikat ACM, lihat [Mengonfigurasi TLS dengan Certificate Manager AWS aktif](#). end-to-end GitHub
  - Hosting Envoy Secret Discovery Service (SDS) — Masukkan nama Utusan rahasia yang akan diambil menggunakan Secret Discovery Service.
  - Hosting file lokal - Tentukan jalur ke file rantai Sertifikat pada sistem file tempat Utusan digunakan. Untuk mengetahui selengkapnya tentang penerapan mesh dengan aplikasi sampel menggunakan enkripsi dengan file lokal, lihat [Mengonfigurasi TLS dengan Sertifikat TLS yang Disediakan File](#) pada. end-to-end GitHub
- (Opsional) Masukkan Nama Alternatif Subjek. Untuk menambahkan tambahan SANs, pilih Tambahkan SAN. SANs harus diformat FQDN atau URI.
- (Opsional) Pilih Berikan sertifikat klien dan salah satu opsi di bawah ini untuk memberikan sertifikat klien saat server memintanya dan mengaktifkan otentikasi TLS bersama. Untuk mempelajari selengkapnya tentang TLS timbal balik, lihat dokumen [Otentikasi TLS Mutual App Mesh](#).

- Hosting Envoy Secret Discovery Service (SDS) — Masukkan nama Utusan rahasia yang akan diambil menggunakan Secret Discovery Service.
  - Hosting file lokal - Tentukan jalur ke file rantai Sertifikat, serta kunci Pribadi, pada sistem file tempat Utusan digunakan.
9. (Opsional) Backend layanan — Tentukan layanan virtual App Mesh yang akan berkomunikasi dengan node virtual.
- Masukkan nama layanan virtual App Mesh atau Nama Sumber Daya Amazon (ARN) lengkap untuk layanan virtual yang berkomunikasi dengan node virtual Anda.
  - (Opsional) Jika Anda ingin mengatur pengaturan TLS unik untuk backend, pilih pengaturan TLS lalu pilih Ganti default.
  - (Opsional) Pilih Terapkan TLS jika Anda ingin meminta node virtual untuk berkomunikasi dengan semua backend menggunakan TLS.
  - (Opsional) Jika Anda hanya ingin meminta penggunaan TLS untuk satu atau lebih port tertentu, maka masukkan nomor di Port. Untuk menambahkan port tambahan, pilih Tambah port. Jika Anda tidak menentukan port apa pun, TLS diberlakukan untuk semua port.
  - Untuk metode Validasi, pilih salah satu opsi berikut. Sertifikat yang Anda tentukan harus sudah ada dan memenuhi persyaratan khusus. Untuk informasi selengkapnya, lihat [Persyaratan sertifikat](#).
    - AWS Private Certificate Authorityhosting — Pilih satu atau lebih Sertifikat yang ada.
    - Hosting Envoy Secret Discovery Service (SDS) — Masukkan nama Utusan rahasia yang akan diambil menggunakan Secret Discovery Service.
    - Hosting file lokal - Tentukan jalur ke file rantai Sertifikat pada sistem file tempat Utusan digunakan.
  - (Opsional) Masukkan Nama Alternatif Subjek. Untuk menambahkan tambahan SANs, pilih Tambahkan SAN. SANsharus diformat FQDN atau URI.
  - (Opsional) Pilih Berikan sertifikat klien dan salah satu opsi di bawah ini untuk memberikan sertifikat klien saat server memintanya dan mengaktifkan otentikasi TLS bersama. Untuk mempelajari selengkapnya tentang TLS timbal balik, lihat dokumen [Otentikasi TLS Mutual](#) App Mesh.
    - Hosting Envoy Secret Discovery Service (SDS) — Masukkan nama Utusan rahasia yang akan diambil menggunakan Secret Discovery Service.

- Hosting file lokal - Tentukan jalur ke file rantai Sertifikat, serta kunci Pribadi, pada sistem file tempat Utusan digunakan.
- Untuk menambahkan backend tambahan, pilih Tambahkan backend.

## 10. (Opsional) Pencatatan

Untuk mengonfigurasi logging, masukkan jalur log akses HTTP yang Anda inginkan untuk digunakan oleh Envoy. Kami merekomendasikan `/dev/stdout` jalur sehingga Anda dapat menggunakan driver log Docker untuk mengekspor log Utusan Anda ke layanan seperti Amazon Logs. CloudWatch

### Note

Log masih harus dicerna oleh agen dalam aplikasi Anda dan dikirim ke tujuan. Jalur file ini hanya menginstruksikan Utusan ke mana harus mengirim log.

## 11. Konfigurasi pendengar

Pendengar mendukung HTTP, HTTP/2GRPC, dan TCP protokol. HTTP tidak didukung.

- a. Jika node virtual Anda mengharapkan lalu lintas masuk, tentukan Port dan Protokol untuk Listener. Pendengar http mengizinkan transisi koneksi ke soket web. Anda dapat mengklik Add Listener untuk menambahkan beberapa pendengar. Tombol Hapus akan menghapus pendengar itu.
- b. (Opsional) Aktifkan kumpulan koneksi

Penggabungan koneksi membatasi jumlah koneksi yang dapat dibuat oleh Utusan secara bersamaan dengan cluster aplikasi lokal. Hal ini dimaksudkan untuk melindungi aplikasi lokal Anda dari kewalahan dengan koneksi dan memungkinkan Anda menyesuaikan pembentukan lalu lintas untuk kebutuhan aplikasi Anda.

Anda dapat mengonfigurasi pengaturan kumpulan koneksi sisi tujuan untuk pendengar simpul virtual. App Mesh menetapkan pengaturan kumpulan koneksi sisi klien ke tak terbatas secara default, menyederhanakan konfigurasi mesh.

### Note


Protokol ConnectionPool dan PortMapping harus sama. Jika protokol listener Anda adalah tcp, tentukan MaxConnections saja. Jika protokol listener

Anda adalah gRPC atau HTTP2, tentukan `maxRequests` saja. Jika protokol listener Anda adalah HTTP, Anda dapat menentukan `MaxConnections` dan `maxPendingRequests`.

- Untuk koneksi maksimum, tentukan jumlah maksimum koneksi keluar.
  - (Opsional) Untuk permintaan tertunda maksimum, tentukan jumlah permintaan yang meluap setelah koneksi maksimum yang akan diantri oleh Utusan. Nilai default-nya adalah 2147483647.
- c. (Opsional) Aktifkan deteksi outlier

Deteksi outlier yang diterapkan pada Utusan klien memungkinkan klien untuk mengambil tindakan segera pada koneksi dengan kegagalan buruk yang diketahui. Ini adalah bentuk implementasi pemutus sirkuit yang melacak status kesehatan masing-masing host di layanan hulu.

Deteksi outlier secara dinamis menentukan apakah titik akhir di cluster hulu berkinerja tidak seperti yang lain dan menghapusnya dari set penyeimbangan beban yang sehat.

 **Note**

Untuk mengonfigurasi deteksi outlier secara efektif untuk server Virtual Node, metode penemuan layanan dari Virtual Node tersebut dapat berupa DNS AWS Cloud Map atau DNS dengan bidang tipe respons yang disetel ke `ENDPOINTS`. Jika Anda menggunakan metode penemuan layanan DNS dengan tipe respons `asLOADBALANCER`, proxy Envoy hanya akan memilih satu alamat IP untuk perutean ke layanan upstream. Ini meniadakan perilaku deteksi outlier untuk mengeluarkan inang yang tidak sehat dari sekumpulan inang. Lihat bagian Metode penemuan Layanan untuk detail selengkapnya tentang perilaku proxy Utusan sehubungan dengan jenis penemuan layanan.


- Untuk kesalahan Server, tentukan jumlah kesalahan 5xx berturut-turut yang diperlukan untuk ejeksi.
- Untuk interval deteksi Outlier, tentukan interval waktu dan unit antara analisis sapuan ejeksi.

- Untuk durasi ejeksi Base, tentukan jumlah dasar waktu dan unit tempat host dikeluarkan.
  - Untuk persentase Ejection, tentukan persentase maksimum host di kolam penyeimbang beban yang dapat dikeluarkan.
- d. (Opsional) Aktifkan pemeriksaan kesehatan — Konfigurasi pengaturan untuk kebijakan pemeriksaan kesehatan.

Kebijakan pemeriksaan kesehatan bersifat opsional, tetapi jika Anda menentukan nilai apa pun untuk kebijakan kesehatan, maka Anda harus menentukan nilai untuk ambang batas Sehat, Interval pemeriksaan Kesehatan, Protokol pemeriksaan Kesehatan, Periode waktu tunggu, dan ambang tidak sehat.

- Untuk protokol pemeriksaan Kesehatan, pilih protokol. Jika Anda memilih `grpc`, maka layanan Anda harus sesuai dengan Protokol Pemeriksaan [Kesehatan GRPC](#).
  - Untuk port pemeriksaan Kesehatan, tentukan port tempat pemeriksaan kesehatan harus dijalankan.
  - Untuk ambang sehat, tentukan jumlah pemeriksaan kesehatan yang berhasil berturut-turut yang harus dilakukan sebelum menyatakan pendengar sehat.
  - Untuk interval pemeriksaan Kesehatan, tentukan periode waktu dalam milidetik antara setiap eksekusi pemeriksaan kesehatan.
  - Untuk Path, tentukan jalur tujuan untuk permintaan pemeriksaan kesehatan. Nilai ini hanya digunakan jika protokol pemeriksaan Kesehatan adalah `http` atau `http2`. Nilai diabaikan untuk protokol lain.
  - Untuk periode Timeout, tentukan jumlah waktu untuk menunggu saat menerima respons dari pemeriksaan kesehatan, dalam milidetik.
  - Untuk ambang tidak sehat, tentukan jumlah pemeriksaan kesehatan gagal berturut-turut yang harus dilakukan sebelum menyatakan pendengar tidak sehat.
- e. (Opsional) Aktifkan penghentian TLS — Konfigurasi bagaimana node virtual lainnya berkomunikasi dengan node virtual ini menggunakan TLS.
- Untuk Mode, pilih mode yang Anda inginkan untuk dikonfigurasi TLS pada pendengar.
  - Untuk metode Sertifikat, pilih salah satu opsi berikut. Sertifikat harus memenuhi persyaratan khusus. Untuk informasi selengkapnya, lihat [Persyaratan sertifikat](#).
    - AWS Certificate Manager hosting — Pilih Sertifikat yang ada.

- Hosting Envoy Secret Discovery Service (SDS) — Masukkan nama Utusan rahasia yang akan diambil menggunakan Secret Discovery Service.
  - Hosting file lokal - Tentukan jalur ke file rantai Sertifikat, serta kunci Pribadi, pada sistem file tempat proxy Utusan digunakan.
  - (Opsional) Pilih Memerlukan sertifikat klien dan salah satu opsi di bawah ini untuk mengaktifkan otentikasi TLS timbal balik ketika klien memberikan sertifikat. Untuk mempelajari selengkapnya tentang TLS timbal balik, lihat dokumen [Otentikasi TLS Mutual](#) App Mesh.
    - Hosting Envoy Secret Discovery Service (SDS) — Masukkan nama Utusan rahasia yang akan diambil menggunakan Secret Discovery Service.
    - Hosting file lokal - Tentukan jalur ke file rantai Sertifikat pada sistem file tempat Utusan digunakan.
  - (Opsional) Masukkan Nama Alternatif Subjek. Untuk menambahkan tambahan SANs, pilih Tambahkan SAN. SANsharus diformat FQDN atau URI.
- f. (Opsional) Timeout

 Note

Jika Anda menentukan batas waktu yang lebih besar dari default, pastikan untuk mengatur router virtual dan rute dengan batas waktu lebih besar dari default. Namun, jika Anda mengurangi batas waktu ke nilai yang lebih rendah dari default, itu opsional untuk memperbarui batas waktu di Route. Untuk informasi selengkapnya, lihat [Rute](#).

- Permintaan batas waktu - Anda dapat menentukan batas waktu permintaan jika Anda memilih grpc, http, atau http2 untuk Protokol pendengar. Defaultnya adalah 15 detik. Nilai 0 menonaktifkan batas waktu.
- Durasi idle - Anda dapat menentukan durasi idle untuk protokol pendengar apa pun. Waktu default adalah 300 detik.

12. Pilih Buat simpul virtual untuk menyelesaikan.

## AWS CLI

Untuk membuat node virtual menggunakan file AWS CLI.

Buat node virtual yang menggunakan DNS untuk penemuan layanan menggunakan perintah berikut dan file JSON input (ganti *red* nilainya dengan milik Anda sendiri):

1. 

```
aws appmesh create-virtual-node \
--cli-input-json file://create-virtual-node-dns.json
```

2. Isi contoh create-virtual-node-dns .json:

```
{
 "meshName": "meshName",
 "spec": {
 "listeners": [
 {
 "portMapping": {
 "port": 80,
 "protocol": "http"
 }
 }
],
 "serviceDiscovery": {
 "dns": {
 "hostname": "serviceBv1.svc.cluster.local"
 }
 }
 },
 "virtualNodeName": "nodeName"
}
```

3. Contoh output:

```
{
 "virtualNode": {
 "meshName": "meshName",
 "metadata": {
 "arn": "arn:aws:appmesh:us-west-2:210987654321:mesh/meshName/
virtualNode/nodeName",
 "createdAt": "2022-04-06T09:12:24.348000-05:00",
 "lastUpdatedAt": "2022-04-06T09:12:24.348000-05:00",
 "meshOwner": "123456789012",
 "resourceOwner": "210987654321",
 "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
 "version": 1
 }
 },
}
```

```
"spec": {
 "listeners": [
 {
 "portMapping": {
 "port": 80,
 "protocol": "http"
 }
 }
],
 "serviceDiscovery": {
 "dns": {
 "hostname": "serviceBv1.svc.cluster.local"
 }
 }
},
"status": {
 "status": "ACTIVE"
},
"virtualNodeName": "nodeName"
}
```

Untuk informasi selengkapnya tentang membuat node virtual dengan App Mesh AWS CLI for, lihat [create-virtual-node](#) perintah di AWS CLI referensi.

## Menghapus simpul virtual

### Note

Anda tidak dapat menghapus node virtual jika ditentukan sebagai target di [route](#) apa pun atau sebagai penyedia di [layanan virtual](#) apa pun.

### Konsol Manajemen AWS

Untuk menghapus node virtual menggunakan Konsol Manajemen AWS

1. Buka konsol App Mesh di <https://console.aws.amazon.com/appmesh/>.
2. Pilih mesh yang ingin Anda hapus dari node virtual. Semua jerat yang Anda miliki dan yang telah [dibagikan](#) dengan Anda terdaftar.

3. Pilih Virtual node di navigasi kiri.
4. Dalam tabel Virtual Nodes, pilih node virtual yang ingin Anda hapus dan pilih Delete. Untuk menghapus node virtual, ID akun Anda harus terdaftar di kolom pemilik Mesh atau pemilik Resource dari node virtual.
5. Di kotak konfirmasi, ketik **delete** lalu pilih Hapus.

## AWS CLI

Untuk menghapus node virtual menggunakan AWS CLI

1. Gunakan perintah berikut untuk menghapus node virtual Anda (ganti *red* nilai dengan milik Anda sendiri):

```
aws appmesh delete-virtual-node \
 --mesh-name meshName \
 --virtual-node-name nodeName
```

2. Contoh output:

```
{
 "virtualNode": {
 "meshName": "meshName",
 "metadata": {
 "arn": "arn:aws:appmesh:us-west-2:210987654321:mesh/meshName/
virtualNode/nodeName",
 "createdAt": "2022-04-06T09:12:24.348000-05:00",
 "lastUpdatedAt": "2022-04-07T11:03:48.120000-05:00",
 "meshOwner": "123456789012",
 "resourceOwner": "210987654321",
 "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
 "version": 2
 },
 "spec": {
 "backends": [],
 "listeners": [
 {
 "portMapping": {
 "port": 80,
 "protocol": "http"
 }
 }
]
 }
 }
}
```

```
],
 "serviceDiscovery": {
 "dns": {
 "hostname": "serviceBv1.svc.cluster.local"
 }
 }
 },
 "status": {
 "status": "DELETED"
 },
 "virtualNodeName": "nodeName"
}
}
```

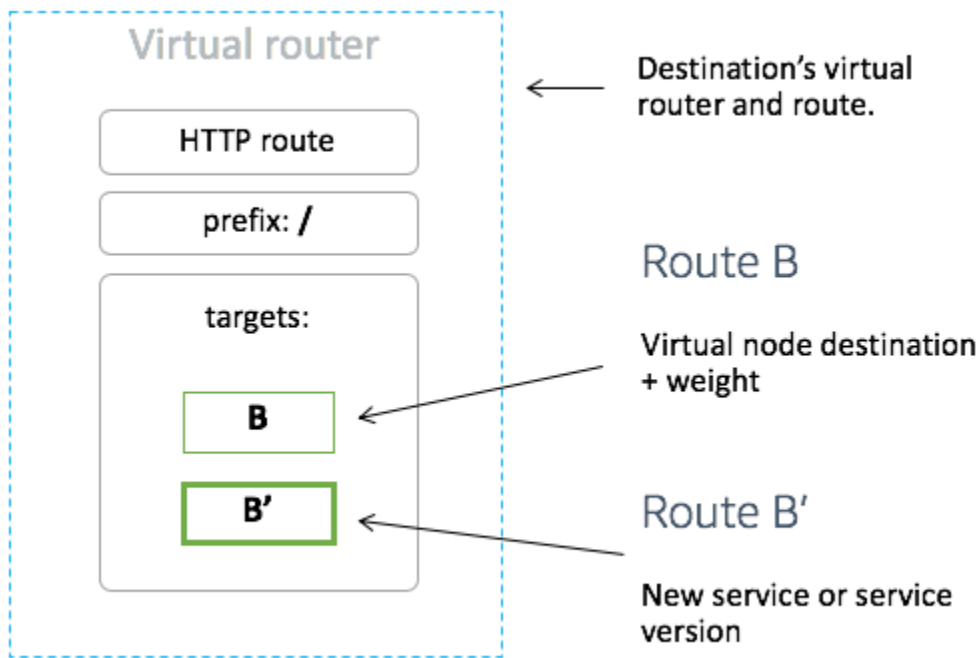
Untuk informasi selengkapnya tentang menghapus node virtual dengan AWS CLI for App Mesh, lihat [delete-virtual-node](#) perintah di AWS CLI referensi.

## Router virtual

### Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk AWS App Mesh. Setelah 30 September 2026, Anda tidak akan lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Router virtual menangani lalu lintas untuk satu atau lebih layanan virtual dalam mesh Anda. Setelah Anda membuat router virtual, Anda dapat membuat dan mengaitkan rute untuk router virtual Anda yang mengarahkan permintaan masuk ke node virtual yang berbeda.



Lalu lintas masuk apa pun yang diharapkan oleh router virtual Anda harus ditentukan sebagai pendengar.

## Membuat router virtual

### Konsol Manajemen AWS

Untuk membuat router virtual menggunakan Konsol Manajemen AWS

#### Note

Saat membuat Router Virtual, Anda harus menambahkan pemilih namespace dengan label untuk mengidentifikasi daftar ruang nama untuk mengaitkan Rute ke Router Virtual yang dibuat.

1. Buka konsol App Mesh di <https://console.aws.amazon.com/appmesh/>.
2. Pilih mesh tempat Anda ingin membuat router virtual. Semua jerat yang Anda miliki dan yang telah [dibagikan](#) dengan Anda terdaftar.
3. Pilih Router virtual di navigasi kiri.
4. Pilih Buat router virtual.

5. Untuk nama router Virtual, tentukan nama untuk router virtual Anda. Hingga 255 huruf, angka, tanda hubung, dan garis bawah diperbolehkan.
6. (Opsional) Untuk konfigurasi Listener, tentukan Port dan Protokol untuk router virtual Anda. `http` Pendengar mengizinkan transisi koneksi ke socket web. Anda dapat mengklik Tambahkan Pendengar untuk menambahkan beberapa pendengar. Tombol Hapus akan menghapus pendengar itu.
7. Pilih Buat router virtual untuk menyelesaikan.

## AWS CLI

Untuk membuat router virtual menggunakan file AWS CLI.

Buat router virtual menggunakan perintah berikut dan masukan JSON (ganti *red* nilainya dengan milik Anda sendiri):

1. 

```
aws appmesh create-virtual-router \
 --cli-input-json file://create-virtual-router.json
```

2. Isi contoh `create-virtual-router.json`

3. 

```
{
 "meshName": "meshName",
 "spec": {
 "listeners": [
 {
 "portMapping": {
 "port": 80,
 "protocol": "http"
 }
 }
]
 },
 "virtualRouterName": "routerName"
}
```

4. Contoh output:

```
{
 "virtualRouter": {
 "meshName": "meshName",
 "metadata": {
```

```
 "arn": "arn:aws:appmesh:us-west-2:210987654321:mesh/meshName/
virtualRouter/routerName",
 "createdAt": "2022-04-06T11:49:47.216000-05:00",
 "lastUpdatedAt": "2022-04-06T11:49:47.216000-05:00",
 "meshOwner": "123456789012",
 "resourceOwner": "210987654321",
 "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
 "version": 1
 },
 "spec": {
 "listeners": [
 {
 "portMapping": {
 "port": 80,
 "protocol": "http"
 }
 }
]
 },
 "status": {
 "status": "ACTIVE"
 },
 "virtualRouterName": "routerName"
}
}
```

Untuk informasi selengkapnya tentang membuat router virtual dengan AWS CLI for App Mesh, lihat [create-virtual-router](#) perintah di AWS CLI referensi.

## Menghapus router virtual

### Note

Anda tidak dapat menghapus router virtual jika memiliki [rule](#) atau jika ditentukan sebagai penyedia untuk [layanan virtual](#) apa pun.

## Konsol Manajemen AWS

Untuk menghapus router virtual menggunakan Konsol Manajemen AWS

1. Buka konsol App Mesh di <https://console.aws.amazon.com/appmesh/>.
2. Pilih mesh yang ingin Anda hapus dari router virtual. Semua jerat yang Anda miliki dan yang telah [dibagikan](#) dengan Anda terdaftar.
3. Pilih Router virtual di navigasi kiri.
4. Di tabel Virtual Routers, pilih router virtual yang ingin Anda hapus dan pilih Delete di sudut kanan atas. Untuk menghapus router virtual, ID akun Anda harus terdaftar di pemilik Mesh atau kolom pemilik Sumber Daya dari router virtual.
5. Di kotak konfirmasi, ketik **delete** dan kemudian klik Hapus.

## AWS CLI

Untuk menghapus router virtual menggunakan AWS CLI

1. Gunakan perintah berikut untuk menghapus router virtual Anda (ganti *red* nilainya dengan milik Anda sendiri):

```
aws appmesh delete-virtual-router \
 --mesh-name meshName \
 --virtual-router-name routerName
```

2. Contoh output:

```
{
 "virtualRouter": {
 "meshName": "meshName",
 "metadata": {
 "arn": "arn:aws:appmesh:us-west-2:210987654321:mesh/meshName/
virtualRouter/routerName",
 "createdAt": "2022-04-06T11:49:47.216000-05:00",
 "lastUpdatedAt": "2022-04-07T10:49:53.402000-05:00",
 "meshOwner": "123456789012",
 "resourceOwner": "210987654321",
 "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
 "version": 2
 },
 "spec": {
```

```
 "listeners": [
 {
 "portMapping": {
 "port": 80,
 "protocol": "http"
 }
 }
],
 "status": {
 "status": "DELETED"
 },
 "virtualRouterName": "routerName"
 }
}
```

Untuk informasi selengkapnya tentang menghapus router virtual dengan AWS CLI for App Mesh, lihat [delete-virtual-router](#) perintah di AWS CLI referensi.

## Rute

### Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk. AWS App Mesh Setelah 30 September 2026, Anda tidak akan lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Rute dikaitkan dengan router virtual. Rute ini digunakan untuk mencocokkan permintaan untuk router virtual dan untuk mendistribusikan lalu lintas ke node virtual yang terkait. Jika rute cocok dengan permintaan, itu dapat mendistribusikan lalu lintas ke satu atau beberapa node virtual target. Anda dapat menentukan bobot relatif untuk setiap node virtual. Topik ini membantu Anda bekerja dengan rute di mesh layanan.

## Membuat rute

### Konsol Manajemen AWS


Untuk membuat rute menggunakan Konsol Manajemen AWS

1. Buka konsol App Mesh di <https://console.aws.amazon.com/appmesh/>.
2. Pilih mesh yang ingin Anda buat rute. Semua jerat yang Anda miliki dan yang telah [dibagikan](#) dengan Anda terdaftar.
3. Pilih Router virtual di navigasi kiri.
4. Pilih router virtual yang ingin Anda kaitkan dengan rute baru. Jika tidak ada yang terdaftar, maka Anda perlu [membuat router virtual](#) terlebih dahulu.
5. Dalam tabel Rute, pilih Buat rute. Untuk membuat rute, ID akun Anda harus terdaftar sebagai pemilik sumber daya rute.
6. Untuk nama Rute, tentukan nama yang akan digunakan untuk rute Anda.
7. Untuk jenis Rute, pilih protokol yang ingin Anda rutekan. Protokol yang Anda pilih harus sesuai dengan protokol listener yang Anda pilih untuk router virtual Anda dan node virtual yang Anda rutekan lalu lintas.
8. (Opsional) Untuk prioritas Rute, tentukan prioritas dari 0-1000 yang akan digunakan untuk rute Anda. Rute dicocokkan berdasarkan nilai yang ditentukan, di mana 0 adalah prioritas tertinggi.
9. (Opsional) Pilih Konfigurasi tambahan. Dari protokol di bawah, pilih protokol yang Anda pilih untuk Jenis rute dan tentukan pengaturan di konsol sesuai keinginan.
10. Untuk konfigurasi Target, pilih node virtual App Mesh yang ada untuk merutekan lalu lintas ke dan menentukan Berat. Anda dapat memilih Tambahkan target untuk menambahkan target tambahan. Persentase untuk semua target harus bertambah hingga 100. Jika tidak ada node virtual yang terdaftar, maka Anda harus [membuatnya](#) terlebih dahulu. Jika node virtual yang dipilih memiliki beberapa pendengar, port Target diperlukan.
11. Untuk konfigurasi Match, tentukan:

Konfigurasi kecocokan tidak tersedia untuk *tcp*

- Jika http/http2 adalah jenis yang dipilih:
  - (Opsional) Metode - menentukan header metode yang akan dicocokkan dalam permintaan http/ http2 masuk.

- (Opsional) Pencocokan port - Cocokkan port untuk lalu lintas masuk. Pencocokan port diperlukan jika router virtual ini memiliki banyak pendengar.
- (Opsional) Prefix/Exact/Regexjalur - metode pencocokan jalur URL.
- Pencocokan awalan - permintaan yang cocok oleh rute gateway ditulis ulang ke nama layanan virtual target dan awalan yang cocok ditulis ulang, secara default. / Bergantung pada bagaimana Anda mengkonfigurasi layanan virtual Anda, itu bisa menggunakan router virtual untuk merutekan permintaan ke node virtual yang berbeda, berdasarkan awalan atau header tertentu.

 Note

Jika Anda mengaktifkan pencocokan berbasis Path/ Prefix, App Mesh mengaktifkan normalisasi jalur ([normalize\\_path](#) dan [merge\\_slashes](#)) untuk meminimalkan kemungkinan kerentanan kebingungan jalur.

Kerentanan kebingungan jalur terjadi ketika pihak yang berpartisipasi dalam permintaan menggunakan representasi jalur yang berbeda.

- Pencocokan persis - param yang tepat menonaktifkan pencocokan sebagian untuk rute dan memastikan bahwa itu hanya mengembalikan rute jika jalurnya sama dengan URL saat ini.
- Pencocokan Regex - digunakan untuk menggambarkan pola di mana beberapa URLs mungkin benar-benar mengidentifikasi satu halaman di situs web.
- (Opsional) Parameter kueri - bidang ini memungkinkan Anda untuk mencocokkan pada parameter kueri.
- (Opsional) Header - menentukan header untuk http dan http2. Itu harus sesuai dengan permintaan yang masuk untuk merutekan ke layanan virtual target..
- Jika grpc adalah tipe yang dipilih:
  - Nama layanan - layanan tujuan yang sesuai dengan permintaan.
  - Nama metode - metode tujuan untuk mencocokkan permintaan.
  - (Opsional) Metadata - menentukan Match berdasarkan keberadaan metadata. Semua harus sesuai untuk permintaan yang akan diproses.

## 12. Pilih Buat rute.

## AWS CLI

Untuk membuat rute menggunakan AWS CLI.

Buat rute gRPC menggunakan perintah berikut dan masukan JSON (ganti *red* nilainya dengan milik Anda sendiri):

- ```
aws appmesh create-route \  
  --cli-input-json file://create-route-grpc.json
```

- Isi contoh create-route-grpc.json

```
{  
  "meshName" : "meshName",  
  "routeName" : "routeName",  
  "spec" : {  
    "grpcRoute" : {  
      "action" : {  
        "weightedTargets" : [  
          {  
            "virtualNode" : "nodeName",  
            "weight" : 100  
          }  
        ]  
      },  
      "match" : {  
        "metadata" : [  
          {  
            "invert" : false,  
            "match" : {  
              "prefix" : "123"  
            },  
            "name" : "myMetadata"  
          }  
        ],  
        "methodName" : "nameOfmethod",  
        "serviceName" : "serviceA.svc.cluster.local"  
      },  
      "retryPolicy" : {  
        "grpcRetryEvents" : [ "deadline-exceeded" ],  
        "httpRetryEvents" : [ "server-error", "gateway-error" ],  
        "maxRetries" : 3,  
        "perRetryTimeout" : {
```

```

        "unit" : "s",
        "value" : 15
      },
      "tcpRetryEvents" : [ "connection-error" ]
    }
  },
  "priority" : 100
},
"virtualRouterName" : "routerName"
}

```

3. Contoh output:

```

{
  "route": {
    "meshName": "meshName",
    "metadata": {
      "arn": "arn:aws:appmesh:us-west-2:210987654321:mesh/meshName/virtualRouter/routerName/route/routeName",
      "createdAt": "2022-04-06T13:48:20.749000-05:00",
      "lastUpdatedAt": "2022-04-06T13:48:20.749000-05:00",
      "meshOwner": "123456789012",
      "resourceOwner": "210987654321",
      "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
      "version": 1
    },
    "routeName": "routeName",
    "spec": {
      "grpcRoute": {
        "action": {
          "weightedTargets": [
            {
              "virtualNode": "nodeName",
              "weight": 100
            }
          ]
        },
        "match": {
          "metadata": [
            {
              "invert": false,
              "match": {
                "prefix": "123"
              }
            }
          ]
        }
      }
    }
  }
}

```

```
        "name": "myMetadata"
      }
    ],
    "methodName": "nameOfMehod",
    "serviceName": "serviceA.svc.cluster.local"
  },
  "retryPolicy": {
"grpcRetryEvents": [
    "deadline-exceeded"
  ],
  "httpRetryEvents": [
    "server-error",
    "gateway-error"
  ],
  "maxRetries": 3,
  "perRetryTimeout": {
    "unit": "s",
    "value": 15
  },
  "tcpRetryEvents": [
    "connection-error"
  ]
}
  },
  "priority": 100
},
"status": {
  "status": "ACTIVE"
},
"virtualRouterName": "routerName"
}
}
```

Untuk informasi selengkapnya tentang membuat rute dengan App Mesh AWS CLI for, lihat perintah [create-route](#) dalam referensi. AWS CLI

gRPC

(Opsional) Pertandingan

- (Opsional) Masukkan nama Layanan dari layanan tujuan agar sesuai dengan permintaan. Jika Anda tidak menentukan nama, permintaan ke layanan apa pun akan dicocokkan.
- (Opsional) Masukkan nama Metode metode tujuan untuk mencocokkan permintaan. Jika Anda tidak menentukan nama, permintaan ke metode apa pun akan dicocokkan. Jika Anda menentukan nama metode, Anda harus menentukan nama layanan.

(Opsional) Metadata

Pilih Tambah Metadata.

- (Opsional) Masukkan nama Metadata yang ingin Anda rute berdasarkan, pilih jenis Match, dan masukkan nilai Match. Memilih Invert akan cocok dengan yang sebaliknya. Misalnya, jika Anda menentukan nama Metadata darimyMetadata, jenis Pencocokan Exact, nilai Match dari123, dan pilih Invert, maka rute dicocokkan untuk setiap permintaan yang memiliki nama metadata yang dimulai dengan apa pun selain. 123
- (Opsional) Pilih Tambahkan metadata untuk menambahkan hingga sepuluh item metadata.

(Opsional) Kebijakan coba lagi

Kebijakan coba lagi memungkinkan klien untuk melindungi diri dari kegagalan jaringan intermiten atau kegagalan sisi server intermiten. Kebijakan coba lagi bersifat opsional, tetapi disarankan. Nilai batas waktu coba lagi menentukan batas waktu per percobaan ulang (termasuk upaya awal). Jika Anda tidak menentukan kebijakan coba lagi, App Mesh dapat secara otomatis membuat kebijakan default untuk setiap rute Anda. Untuk informasi selengkapnya, lihat [Kebijakan coba lagi rute default](#).

- Untuk Timeout Coba lagi, masukkan jumlah unit untuk durasi batas waktu. Nilai diperlukan jika Anda memilih acara coba ulang protokol apa pun.
- Untuk unit batas waktu Coba lagi, pilih unit. Nilai diperlukan jika Anda memilih acara coba ulang protokol apa pun.
- Untuk percobaan ulang Max, masukkan jumlah maksimum percobaan ulang saat permintaan gagal. Nilai diperlukan jika Anda memilih acara coba ulang protokol apa pun. Kami merekomendasikan nilai setidaknya dua.

- Pilih satu atau beberapa acara coba lagi HTTP. Sebaiknya pilih setidaknya `stream-error` dan `gateway-error`.
- Pilih acara coba lagi TCP.
- Pilih satu atau beberapa acara coba lagi gRPC. Kami menyarankan untuk memilih setidaknya `dibatalkan` dan `tidak tersedia`.

(Opsional) Timeout

- Defaultnya adalah 15 detik. Jika Anda menetapkan kebijakan Coba Ulang, maka durasi yang Anda tentukan di sini harus selalu lebih besar dari atau sama dengan durasi percobaan ulang dikalikan dengan percobaan ulang Maks yang Anda tetapkan dalam kebijakan Coba Ulang sehingga kebijakan coba ulang Anda dapat selesai. Jika Anda menentukan durasi lebih dari 15 detik, maka pastikan batas waktu yang ditentukan untuk pendengar Target node virtual mana pun juga lebih besar dari 15 detik. Untuk informasi selengkapnya, lihat [Virtual Nodes](#).
- Nilai `0` menonaktifkan batas waktu.
- Jumlah maksimum waktu rute dapat menganggur.

HTTP dan HTTP/2

(Opsional) Pertandingan

- Tentukan Awalan bahwa rute harus cocok. Misalnya, jika nama layanan virtual Anda `service-b.local` dan Anda menginginkan agar rute cocok dengan permintaan `service-b.local/metrics`, prefiks anda seharusnya `/metrics`. Menentukan `/` rute semua lalu lintas.
- (Opsional) Pilih Metode.
- (Opsional) Pilih Skema. Hanya berlaku untuk HTTP2 rute.

(Opsional) Header

- (Opsional) Pilih Tambahkan header. Masukkan nama Header yang ingin Anda rute berdasarkan, pilih jenis Match, dan masukkan nilai Match. Memilih Invert akan cocok dengan yang sebaliknya. Misalnya, jika Anda menentukan header bernama `clientRequestId` dengan Awalan dari `123`, dan pilih Invert, maka rute dicocokkan untuk setiap permintaan yang memiliki header yang dimulai dengan apa pun selain. `123`
- (Opsional) Pilih Tambahkan header. Anda dapat menambahkan hingga sepuluh header.

(Opsional) Kebijakan coba lagi

Kebijakan coba lagi memungkinkan klien untuk melindungi diri dari kegagalan jaringan intermiten atau kegagalan sisi server intermiten. Kebijakan coba lagi bersifat opsional, tetapi disarankan. Nilai batas waktu coba lagi menentukan batas waktu per percobaan ulang (termasuk upaya awal). Jika Anda tidak menentukan kebijakan coba lagi, App Mesh dapat secara otomatis membuat kebijakan default untuk setiap rute Anda. Untuk informasi selengkapnya, lihat [Kebijakan coba lagi rute default](#).

- Untuk Timeout Coba lagi, masukkan jumlah unit untuk durasi batas waktu. Nilai diperlukan jika Anda memilih acara coba ulang protokol apa pun.
- Untuk unit batas waktu Coba lagi, pilih unit. Nilai diperlukan jika Anda memilih acara coba ulang protokol apa pun.
- Untuk percobaan ulang Max, masukkan jumlah maksimum percobaan ulang saat permintaan gagal. Nilai diperlukan jika Anda memilih acara coba ulang protokol apa pun. Kami merekomendasikan nilai setidaknya dua.
- Pilih satu atau beberapa acara coba lagi HTTP. Sebaiknya pilih setidaknya stream-error dan gateway-error.
- Pilih acara coba lagi TCP.

(Opsional) Timeout

- Permintaan batas waktu - Standarnya adalah 15 detik. Jika Anda menetapkan kebijakan Coba Ulang, maka durasi yang Anda tentukan di sini harus selalu lebih besar dari atau sama dengan durasi percobaan ulang dikalikan dengan percobaan ulang Maks yang Anda tetapkan dalam kebijakan Coba Ulang sehingga kebijakan coba ulang Anda dapat selesai.
- Durasi idle - Defaultnya adalah 300 detik.
- Nilai 0 menonaktifkan batas waktu.

Note

Jika Anda menentukan batas waktu yang lebih besar dari default, pastikan batas waktu yang ditentukan untuk pendengar untuk semua peserta node virtual juga lebih besar dari default. Namun, jika Anda mengurangi batas waktu ke nilai yang lebih rendah dari default, itu opsional untuk memperbarui batas waktu di node virtual. Untuk informasi selengkapnya, lihat [Virtual Nodes](#).

TCP

(Opsional) Timeout

- Durasi idle - Defaultnya adalah 300 detik.
- Nilai 0 menonaktifkan batas waktu.

Menghapus rute

Konsol Manajemen AWS

Untuk menghapus rute menggunakan Konsol Manajemen AWS

1. Buka konsol App Mesh di <https://console.aws.amazon.com/appmesh/>.
2. Pilih mesh dari mana Anda ingin menghapus rute. Semua jerat yang Anda miliki dan yang telah [dibagikan](#) dengan Anda terdaftar.
3. Pilih Router virtual di navigasi kiri.
4. Pilih router dari mana Anda ingin menghapus rute.
5. Dalam tabel Rute, pilih rute yang ingin Anda hapus dan pilih Hapus di sudut kanan atas.
6. Di kotak konfirmasi, ketik **delete** dan kemudian klik Hapus.

AWS CLI

Untuk menghapus rute menggunakan AWS CLI

1. Gunakan perintah berikut untuk menghapus rute Anda (ganti *red* nilainya dengan milik Anda):

```
aws appmesh delete-route \  
  --mesh-name meshName \  
  --virtual-router-name routerName \  
  --route-name routeName
```

2. Contoh output:

```
{  
  "route": {  
    "meshName": "meshName",  
    "metadata": {
```

```
    "arn": "arn:aws:appmesh:us-west-2:210987654321:mesh/meshName/  
virtualRouter/routerName/route/routeName",  
    "createdAt": "2022-04-06T13:46:54.750000-05:00",  
    "lastUpdatedAt": "2022-04-07T10:43:57.152000-05:00",  
    "meshOwner": "123456789012",  
    "resourceOwner": "210987654321",  
    "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",  
    "version": 2  
  },  
  "routeName": "routeName",  
  "spec": {  
    "grpcRoute": {  
      "action": {  
        "weightedTargets": [  
          {  
            "virtualNode": "nodeName",  
            "weight": 100  
          }  
        ]  
      },  
      "match": {  
        "metadata": [  
          {  
            "invert": false,  
            "match": {  
              "prefix": "123"  
            },  
            "name": "myMetadata"  
          }  
        ],  
        "methodName": "methodName",  
        "serviceName": "serviceA.svc.cluster.local"  
      },  
      "retryPolicy": {  
        "grpcRetryEvents": [  
          "deadline-exceeded"  
        ],  
        "httpRetryEvents": [  
          "server-error",  
          "gateway-error"  
        ],  
        "maxRetries": 3,  
        "perRetryTimeout": {  
          "unit": "s",
```

```
        "value": 15
      },
      "tcpRetryEvents": [
        "connection-error"
      ]
    }
  },
  "priority": 100
},
"status": {
  "status": "DELETED"
},
"virtualRouterName": "routerName"
}
}
```

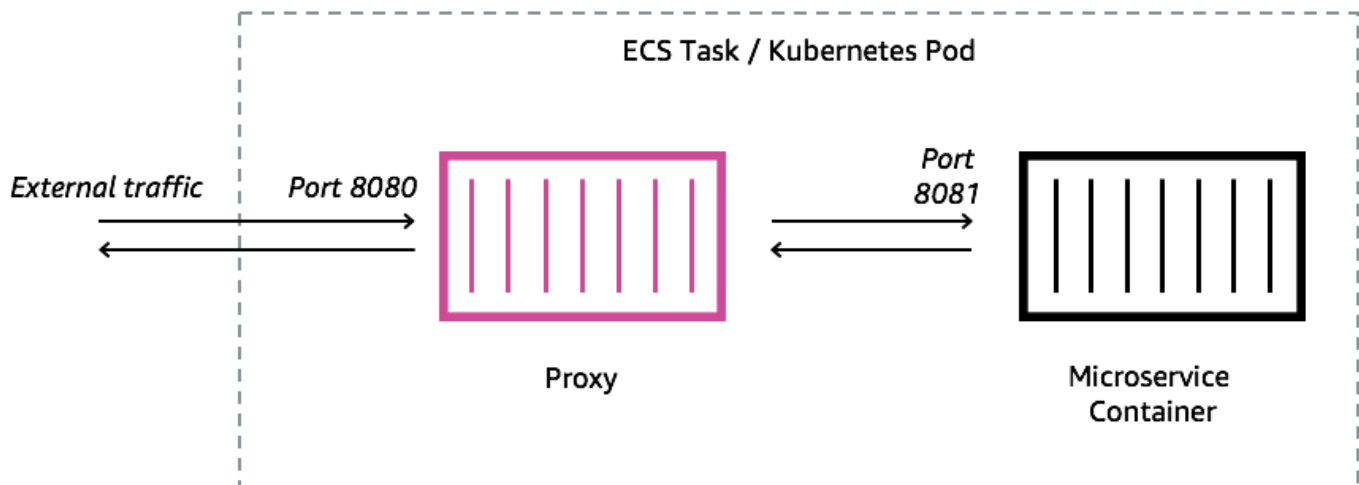
Untuk informasi selengkapnya tentang menghapus rute dengan App Mesh AWS CLI for, lihat perintah [delete-route](#) dalam referensi. AWS CLI

Gambar utusan

⚠ Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk AWS App Mesh. Setelah 30 September 2026, Anda tidak lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

AWS App Mesh adalah mesh layanan berdasarkan proxy [Utusan](#).



Anda harus menambahkan proxy Envoy ke tugas Amazon ECS, pod Kubernetes, atau instans Amazon EC2 yang diwakili oleh endpoint App Mesh Anda, seperti node virtual atau gateway virtual. App Mesh menjual gambar kontainer proxy Envoy yang ditambah dengan kerentanan dan pembaruan kinerja terbaru. App Mesh menguji setiap rilis proxy Envoy baru terhadap set fitur App Mesh sebelum membuat gambar baru tersedia untuk Anda.

Varian gambar utusan

App Mesh menyediakan dua varian image container proxy Envoy. Perbedaan antara keduanya adalah bagaimana proxy Envoy berkomunikasi ke bidang data App Mesh dan bagaimana

proxy Utusan berkomunikasi satu sama lain. Salah satunya adalah gambar standar, yang berkomunikasi dengan titik akhir layanan App Mesh standar. Varian lainnya sesuai dengan FIPS, yang berkomunikasi dengan titik akhir layanan App Mesh FIPS dan memberlakukan kriptografi FIPS dalam komunikasi TLS antara layanan App Mesh.

Anda dapat memilih salah satu gambar Regional dari daftar di bawah ini atau gambar dari [repositori publik](#) kami bernama. `aws-appmesh-envoy`

Important

- Mulai 30 Juni 2023, hanya gambar Envoy `v1.17.2.0-prod` atau yang lebih baru yang kompatibel untuk digunakan dengan App Mesh. Untuk pelanggan saat ini yang menggunakan gambar Utusan sebelumnya `v1.17.2.0`, meskipun utusan yang ada akan terus kompatibel, kami sangat menyarankan untuk bermigrasi ke versi terbaru.
- Sebagai praktik terbaik, meningkatkan versi Utusan ke versi terbaru secara teratur sangat disarankan. Hanya versi Utusan terbaru yang divalidasi dengan patch keamanan terbaru, rilis fitur, dan peningkatan kinerja.
- Versi `1.17` adalah pembaruan signifikan untuk Utusan. Lihat [Memperbarui/Migrasi ke Utusan 1.17 untuk detail selengkapnya](#).
- Versi `1.20.0.1` atau yang lebih baru ARM64 kompatibel.
- Untuk IPv6 dukungan, versi Utusan `1.20` atau yang lebih baru diperlukan.

Note

FIPS hanya tersedia di Wilayah yang ditemukan di AS dan Kanada.

Semua Wilayah yang [didukung](#) dapat diganti *Region-code* dengan Wilayah apa pun selain `me-south-1,ap-east-1,ap-southeast-3,,eu-south-1,il-central-1,danf-south-1`.

Standar

```
840364872350.dkr.ecr.region-code.amazonaws.com/aws-appmesh-envoy:v1.34.13.0-prod
```

Sesuai FIPS

```
840364872350.dkr.ecr.region-code.amazonaws.com/aws-appmesh-envoy:v1.34.13.0-prod-  
fips
```

me-south-1

Standar

```
772975370895.dkr.ecr.me-south-1.amazonaws.com/aws-appmesh-envoy:v1.34.13.0-prod
```

ap-east-1

Standar

```
856666278305.dkr.ecr.ap-east-1.amazonaws.com/aws-appmesh-envoy:v1.34.13.0-prod
```

ap-southeast-3

Standar

```
909464085924.dkr.ecr.ap-southeast-3.amazonaws.com/aws-appmesh-envoy:v1.34.13.0-prod
```

eu-south-1

Standar

```
422531588944.dkr.ecr.eu-south-1.amazonaws.com/aws-appmesh-envoy:v1.34.13.0-prod
```

il-central-1

Standar

```
564877687649.dkr.ecr.il-central-1.amazonaws.com/aws-appmesh-envoy:v1.34.13.0-prod
```

af-south-1

Standar

```
924023996002.dkr.ecr.af-south-1.amazonaws.com/aws-appmesh-envoy:v1.34.13.0-prod
```

Public repository

Standar

```
public.ecr.aws/appmesh/aws-appmesh-envoy:v1.34.13.0-prod
```

Sesuai FIPS

```
public.ecr.aws/appmesh/aws-appmesh-envoy:v1.34.13.0-prod-fips
```

Note

Kami merekomendasikan mengalokasikan 512 unit CPU dan setidaknya 64 MiB memori ke wadah Envoy. Di Fargate jumlah memori terendah yang dapat Anda atur adalah 1024 MiB memori. Alokasi sumber daya ke wadah Envoy dapat ditingkatkan jika wawasan kontainer atau metrik lainnya menunjukkan sumber daya yang tidak mencukupi karena beban yang lebih tinggi.

Note

Semua versi rilis `aws-appmesh-envoy` gambar mulai dari `v1.22.0.0` dibangun sebagai gambar Docker distroless. Kami membuat perubahan ini sehingga kami dapat mengurangi ukuran gambar dan mengurangi eksposur kerentanan kami dalam paket yang tidak digunakan yang ada dalam gambar. Jika Anda membangun di atas `aws-appmesh-envoy` gambar dan mengandalkan beberapa paket AL2 dasar (misalnya yum) dan fungsionalitas, maka kami sarankan Anda menyalin binari dari dalam `aws-appmesh-envoy` gambar untuk membangun gambar Docker baru dengan basis AL2

Jalankan skrip ini untuk menghasilkan gambar docker khusus dengan tag `aws-appmesh-envoy:v1.22.0.0-prod-al2`:

```
cat << EOF > Dockerfile
FROM public.ecr.aws/appmesh/aws-appmesh-envoy:v1.22.0.0-prod as envoy

FROM public.ecr.aws/amazonlinux/amazonlinux:2
RUN yum -y update && \
    yum clean all && \
    rm -rf /var/cache/yum
```

```
COPY --from=envoy /usr/bin/envoy /usr/bin/envoy
COPY --from=envoy /usr/bin/agent /usr/bin/agent
COPY --from=envoy /aws_appmesh_aggregate_stats.wasm /
aws_appmesh_aggregate_stats.wasm

CMD [ "/usr/bin/agent" ]
EOF

docker build -f Dockerfile -t aws-appmesh-envoy:v1.22.0.0-prod-al2 .
```

Akses ke gambar kontainer ini di Amazon ECR dikendalikan oleh AWS Identity and Access Management (IAM). Akibatnya, Anda harus menggunakan IAM untuk memverifikasi bahwa Anda telah membaca akses ke Amazon ECR. Misalnya, saat menggunakan Amazon ECS, Anda dapat menetapkan peran eksekusi tugas yang sesuai ke tugas Amazon ECS. Jika Anda menggunakan kebijakan IAM yang membatasi akses ke sumber daya Amazon ECR tertentu, pastikan untuk memverifikasi bahwa Anda mengizinkan akses ke Nama Sumber Daya Amazon (ARN) khusus Wilayah yang mengidentifikasi repositori. `aws-appmesh-envoy` Misalnya, di `us-west-2` Wilayah, Anda mengizinkan akses ke sumber daya berikut: `arn:aws:ecr:us-west-2:840364872350:repository/aws-appmesh-envoy`. Untuk informasi selengkapnya, lihat [Kebijakan Terkelola Amazon ECR](#). Jika Anda menggunakan Docker pada instans Amazon EC2, autentikasi Docker ke repositori. Untuk informasi lebih lanjut, lihat [Autentikasi Registri](#).

Kami terkadang merilis fitur App Mesh baru yang bergantung pada perubahan Utusan yang belum digabungkan ke gambar Utusan hulu. Untuk menggunakan fitur App Mesh baru ini sebelum perubahan Utusan digabungkan di hulu, Anda harus menggunakan image container Envoy yang dijual App Mesh. Untuk daftar perubahan, lihat [masalah GitHub peta jalan App Mesh](#) dengan Envoy Upstream label. Sebaiknya gunakan image container App Mesh Envoy sebagai opsi terbaik yang didukung.

Variabel konfigurasi utusan

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk. AWS App Mesh Setelah 30 September 2026, Anda tidak lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi

lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Gunakan variabel lingkungan berikut untuk mengonfigurasi container Envoy untuk grup tugas node virtual App Mesh Anda.

Note

App Mesh Envoy 1.17 tidak mendukung API v2 xDs Envoy. Jika Anda menggunakan [variabel konfigurasi Envoy yang menerima file konfigurasi](#) Envoy, mereka harus diperbarui ke API xDs v3 terbaru.

Variabel yang dibutuhkan

Variabel lingkungan berikut diperlukan untuk semua container App Mesh Envoy. Variabel ini hanya dapat digunakan dengan versi 1.15.0 atau yang lebih baru dari gambar Utusan. Jika Anda menggunakan versi gambar yang lebih lama, maka Anda harus mengatur APPMESH_VIRTUAL_NODE_NAME variabel sebagai gantinya.

APPMESH_RESOURCE_ARN

Saat Anda menambahkan wadah Envoy ke grup tugas, atur variabel lingkungan ini ke ARN dari node virtual atau gateway virtual yang diwakili oleh grup tugas. Daftar berikut berisi contoh ARNs:

- Node virtual - `arn:aws:appmesh: ::mesh//virtualNode/ Region-code 111122223333 meshName virtualNodeName`
- Gerbang virtual — `arn:aws:appmesh: ::mesh//virtualGateway/ Region-code 111122223333 meshName virtualGatewayName`

Variabel opsional

Variabel lingkungan berikut adalah opsional untuk wadah App Mesh Envoy.

ENVOY_LOG_LEVEL

Menentukan tingkat log untuk wadah Utusan.

Nilai yang valid: `trace,debug,info,warn,error,critical, off`

Default: `info`

ENVOY_INITIAL_FETCH_TIMEOUT

Menentukan jumlah waktu Utusan menunggu respons konfigurasi pertama dari server manajemen selama proses inisialisasi.

Untuk informasi selengkapnya, lihat [Sumber konfigurasi di Dokumentasi](#) Utusan. Ketika diatur ke `0`, tidak ada batas waktu.

Default: `0`

ENVOY_CONCURRENCY

Menetapkan opsi baris `--concurrency` perintah saat memulai Utusan. Ini tidak diatur secara default. Opsi ini tersedia dari versi Utusan `v1.24.0.0-prod` atau di atasnya.

Untuk informasi selengkapnya, lihat [Opsi baris perintah di Dokumentasi](#) Utusan.

Variabel admin

Gunakan variabel lingkungan ini untuk mengkonfigurasi antarmuka administratif Envoy.

ENVOY_ADMIN_ACCESS_PORT

Tentukan port admin khusus untuk didengarkan oleh Envoy. Default: `9901`.

Note

Port admin Envoy harus berbeda dari port pendengar mana pun di gateway virtual atau node virtual

ENVOY_ADMIN_ACCESS_LOG_FILE

Tentukan jalur khusus untuk menulis log akses Utusan ke. Default: `/tmp/envoy_admin_access.log`.

ENVOY_ADMIN_ACCESS_ENABLE_IPV6

Beralih antarmuka administrasi Envoy untuk menerima IPv6 lalu lintas, yang memungkinkan antarmuka ini menerima keduanya dan lalu lintas. IPv4 IPv6 Secara default flag ini disetel ke

false, dan Envoy hanya mendengarkan lalu lintas. IPv4 Variabel ini hanya dapat digunakan dengan Envoy image versi 1.22.0 atau yang lebih baru.

Variabel agen

Gunakan variabel lingkungan ini untuk mengkonfigurasi AWS App Mesh Agen untuk Utusan. Untuk informasi selengkapnya, lihat [Agen App Mesh untuk Utusan](#).

APPNET_ENVOY_RESTART_COUNT

Menentukan berapa kali Agen memulai ulang proses proxy Envoy dalam tugas atau pod yang sedang berjalan jika keluar. Agen juga mencatat status keluar setiap kali Utusan keluar untuk memudahkan pemecahan masalah. Nilai default dari variabel ini adalah 0. Ketika nilai default ditetapkan, Agen tidak mencoba untuk memulai ulang proses.

Default: 0

Maksimum: 10

PID_POLL_INTERVAL_MS

Menentukan interval dalam milidetik di mana status proses proxy Utusan diperiksa oleh Agen. Nilai default-nya adalah 100.

Default: 100

Minimal: 100

Maksimum: 1000

LISTENER_DRAIN_WAIT_TIME_S

Menentukan jumlah waktu dalam hitungan detik proxy Utusan menunggu koneksi aktif ditutup sebelum proses keluar.

Default: 20

Minimal: 5

Maksimum: 110

APPNET_AGENT_ADMIN_MODE

Memulai server antarmuka manajemen Agen dan mengikatnya ke alamat tcp atau socket unix.

Nilai valid: tcp, uds

APPNET_AGENT_HTTP_PORT

Tentukan port yang akan digunakan untuk mengikat antarmuka manajemen Agen dalam tcp mode. Pastikan nilai port adalah > 1024 jika uid != 0. Pastikan port kurang dari 65535.

Default: 9902

APPNET_AGENT_ADMIN_UDS_PATH

Tentukan jalur soket domain unix untuk antarmuka manajemen Agen dalam uds mode.

Default: /var/run/ecs/appnet_admin.sock

Menelusuri variabel

Anda dapat mengonfigurasi tidak satu pun atau salah satu driver penelusuran berikut.

AWS X-Ray variabel

Gunakan variabel lingkungan berikut untuk mengonfigurasi App Mesh dengan AWS X-Ray. Lihat informasi selengkapnya di [Panduan Developer AWS X-Ray](#).

ENABLE_ENVOY_XRAY_TRACING

Mengaktifkan penelusuran X-Ray menggunakan 127.0.0.1:2000 sebagai titik akhir daemon default. Untuk mengaktifkan, atur nilainya ke 1. Nilai default-nya adalah 0.

XRAY_DAEMON_PORT

Tentukan nilai port untuk mengganti port daemon X-Ray default: 2000

XRAY_SAMPLING_RATE

Tentukan laju pengambilan sampel untuk mengganti laju pengambilan sampel default pelacak X-Ray sebesar (5%). 0.05 Tentukan nilai sebagai desimal antara 0 dan 1.00 (100%). Nilai ini diganti jika XRAY_SAMPLING_RULE_MANIFEST ditentukan. Variabel ini didukung dengan gambar Envoy versi v1.19.1.1-prod dan yang lebih baru.

XRAY_SAMPLING_RULE_MANIFEST

Tentukan jalur file di sistem file wadah Envoy untuk mengonfigurasi aturan pengambilan sampel khusus yang dilokalkan untuk pelacak X-Ray. Untuk informasi selengkapnya, lihat [Aturan](#)

[pengambilan sampel](#) di Panduan AWS X-Ray Pengembang. Variabel ini didukung dengan gambar Envoy versi `v1.19.1.0-prod` dan yang lebih baru.

XRAY_SEGMENT_NAME

Tentukan nama segmen untuk jejak untuk mengganti nama segmen X-Ray default. Secara default nilai ini akan ditetapkan sebagai `mesh/resourceName`. Variabel ini didukung dengan versi gambar Envoy `v1.23.1.0-prod` atau yang lebih baru.

Variabel penelusuran Datadog

Variabel lingkungan berikut membantu Anda mengonfigurasi App Mesh dengan pelacak agen Datadog. Untuk informasi selengkapnya, lihat [Konfigurasi Agen](#) dalam dokumentasi Datadog.

ENABLE_ENVOY_DATADOG_TRACING

Mengaktifkan pengumpulan jejak Datadog menggunakan `127.0.0.1:8126` sebagai titik akhir agen Datadog default. Untuk mengaktifkan, atur nilainya ke `1` (nilai default adalah `0`).

DATADOG_TRACER_PORT

Tentukan nilai port untuk mengganti port agen Datadog default: `8126`

DATADOG_TRACER_ADDRESS

Tentukan alamat IP untuk mengganti alamat agen Datadog default: `127.0.0.1`

DD_SERVICE

Tentukan nama layanan untuk jejak untuk mengganti nama layanan Datadog default: `/envoy-meshName virtualNodeName`. Variabel ini didukung dengan gambar Envoy versi `v1.18.3.0-prod` dan yang lebih baru.

Variabel penelusuran Jaeger

Gunakan variabel lingkungan berikut untuk mengonfigurasi App Mesh dengan penelusuran Jaeger. Untuk informasi selengkapnya, lihat [Memulai](#) di dokumentasi Jaeger. Variabel-variabel ini didukung dengan gambar Envoy versi `1.16.1.0-prod` dan yang lebih baru.

ENABLE_ENVOY_JAEGER_TRACING

Mengaktifkan koleksi jejak Jaeger menggunakan `127.0.0.1:9411` sebagai titik akhir Jaeger default. Untuk mengaktifkan, atur nilainya ke `1` (nilai default adalah `0`).

JAEGER_TRACER_PORT

Tentukan nilai port untuk mengganti port Jaeger default: 9411

JAEGER_TRACER_ADDRESS

Tentukan alamat IP untuk mengganti alamat Jaeger default: 127.0.0.1

JAEGER_TRACER_VERSION

Tentukan apakah kolektor membutuhkan jejak dalam JSON atau format yang PROTO dikodekan. Secara default nilai ini akan diatur kePROTO. Variabel ini didukung dengan versi gambar Envoy v1.23.1.0-prod atau yang lebih baru.

Variabel penelusuran utusan

Tetapkan variabel lingkungan berikut untuk menggunakan konfigurasi penelusuran Anda sendiri.

ENVOY_TRACING_CFG_FILE

Tentukan jalur file di sistem file wadah Utusan. Untuk informasi selengkapnya, lihat [config.trace.v3.Tracing](#) di dokumentasi Utusan.

Note

Jika konfigurasi penelusuran memerlukan penentuan cluster penelusuran, pastikan untuk mengonfigurasi konfigurasi klaster terkait di bawah `static_resources` dalam file konfigurasi penelusuran yang sama. Misalnya, Zipkin memiliki [collector_cluster](#) bidang untuk nama cluster yang menampung kolektor jejak, dan cluster itu perlu didefinisikan secara statis.

DogStatsD variabel

Gunakan variabel lingkungan berikut untuk mengonfigurasi App Mesh dengan DogStats D. Untuk informasi lebih lanjut, lihat dokumentasi [DogStatsD](#).

ENABLE_ENVOY_DOG_STATSD

Mengaktifkan statistik DogStats D menggunakan 127.0.0.1:8125 sebagai titik akhir daemon default. Untuk mengaktifkan, atur nilainya ke1.

STATSD_PORT

Tentukan nilai port untuk mengganti port daemon DogStats D default.

STATSD_ADDRESS

Tentukan nilai alamat IP untuk mengganti alamat IP daemon DogStats D default. Default: 127.0.0.1. Variabel ini hanya dapat digunakan dengan versi 1.15.0 atau yang lebih baru dari gambar Utusan.

STATSD_SOCKET_PATH

Tentukan soket domain unix untuk daemon DogStats D. Jika variabel ini tidak ditentukan dan DogStats D diaktifkan, maka nilai ini default ke port alamat IP daemon DogStats D dari 127.0.0.1:8125. Jika ENV0Y_STATS_SINKS_CFG_FILE variabel ditentukan berisi konfigurasi stats sink, itu mengesampingkan semua variabel D. DogStats Variabel ini didukung dengan versi gambar Envoy v1.19.1.0-prod atau yang lebih baru.

Variabel App Mesh

Variabel berikut membantu Anda mengonfigurasi App Mesh.

APPMESH_RESOURCE_CLUSTER

Secara default, App Mesh menggunakan nama sumber daya yang Anda tentukan APPMESH_RESOURCE_ARN saat Envoy merujuk dirinya sendiri dalam metrik dan jejak. Anda dapat menimpa perilaku ini dengan mengatur variabel lingkungan APPMESH_RESOURCE_CLUSTER dengan nama Anda sendiri. Variabel ini hanya dapat digunakan dengan versi 1.15.0 atau yang lebih baru dari gambar Utusan.

APPMESH_METRIC_EXTENSION_VERSION

Tetapkan nilainya 1 untuk mengaktifkan ekstensi metrik App Mesh. Untuk informasi selengkapnya tentang menggunakan ekstensi metrik App Mesh, lihat [Ekstensi metrik untuk App Mesh](#).

APPMESH_DUALSTACK_ENDPOINT

Tetapkan nilai untuk terhubung 1 ke titik akhir App Mesh Dual Stack. Saat flag ini disetel, Envoy menggunakan domain berkemampuan tumpukan ganda kami. Secara default flag ini diatur ke false dan hanya terhubung ke IPv4 domain kami. Variabel ini hanya dapat digunakan dengan Envoy image versi 1.22.0 atau yang lebih baru.

Variabel statistik utusan

Gunakan variabel lingkungan berikut untuk mengonfigurasi App Mesh dengan Statistik Utusan. Untuk informasi selengkapnya, lihat dokumentasi [Statistik Utusan](#).

ENABLE_ENVOY_STATS_TAGS

Mengaktifkan penggunaan tag yang ditentukan App Mesh `appmesh.mesh` dan `appmesh.virtual_node`. Untuk informasi selengkapnya, lihat [config.metrics.v3.TagSpecifier](#) dalam dokumentasi Utusan. Untuk mengaktifkan, atur nilainya ke 1.

ENVOY_STATS_CONFIG_FILE

Tentukan jalur file di sistem file kontainer Envoy untuk mengganti file konfigurasi tag Statistik default dengan file Anda sendiri. Untuk informasi selengkapnya, lihat [config.metrics.v3.StatsConfig](#).

Note

Menyetel konfigurasi statistik khusus yang menyertakan filter statistik dapat menyebabkan Envoy memasuki status yang tidak lagi disinkronkan dengan benar dengan status App Mesh di dunia. Ini adalah [bug](#) di Envoy. Rekomendasi kami adalah untuk tidak melakukan penyaringan statistik di Envoy. Jika pemfilteran mutlak diperlukan, kami telah mencantumkan beberapa solusi dalam [masalah](#) ini di peta jalan kami.

ENVOY_STATS_SINKS_CFG_FILE

Tentukan jalur file di sistem file kontainer Envoy untuk mengganti konfigurasi default dengan konfigurasi Anda sendiri. Untuk informasi selengkapnya, lihat [config.metrics.v3.StatsSink](#) dalam dokumentasi Utusan.

Variabel usang

Variabel lingkungan `APPMESH_VIRTUAL_NODE_NAME` dan `APPMESH_RESOURCE_NAME` tidak lagi didukung dalam versi Envoy 1.15.0 atau yang lebih baru. Namun, mereka masih didukung untuk jerat yang ada. Alih-alih menggunakan variabel ini dengan versi Envoy 1.15.0 atau yang lebih baru, gunakan `APPMESH_RESOURCE_ARN` untuk semua titik akhir App Mesh.

Default utusan ditetapkan oleh App Mesh

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk AWS App Mesh. Setelah 30 September 2026, Anda tidak lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Bagian berikut memberikan informasi tentang default Envoy untuk kebijakan coba ulang rute dan pemutus sirkuit yang ditetapkan oleh App Mesh.

Kebijakan coba lagi rute default

Jika Anda tidak memiliki jerat di akun Anda sebelum 29 Juli 2020, App Mesh secara otomatis membuat kebijakan coba lagi rute Utusan default untuk semua permintaan HTTP, HTTP/2, dan gRPC di mesh apa pun di akun Anda pada atau setelah 29 Juli 2020. Jika Anda memiliki jerat di akun Anda sebelum 29 Juli 2020, maka tidak ada kebijakan default yang dibuat untuk rute Utusan apa pun yang ada sebelum, pada, atau setelah 29 Juli 2020. Ini kecuali Anda [membuka tiket dengan AWS dukungan](#). Setelah dukungan memproses tiket, kebijakan default dibuat untuk rute Envoy masa depan yang dibuat App Mesh pada atau setelah tanggal tiket diproses. [Untuk informasi selengkapnya tentang kebijakan coba lagi rute Utusan, lihat config.route.v3. RetryPolicy](#) dalam dokumentasi Utusan.

App Mesh membuat rute Envoy saat Anda membuat [rute](#) App Mesh atau menentukan penyedia node virtual untuk layanan [virtual](#) App Mesh. Meskipun Anda dapat membuat kebijakan percobaan ulang rute App Mesh, Anda tidak dapat membuat kebijakan coba lagi App Mesh untuk penyedia node virtual.

Kebijakan default tidak terlihat melalui App Mesh API. Kebijakan default hanya dapat dilihat melalui Envoy. Untuk melihat konfigurasi, [aktifkan antarmuka administrasi](#) dan kirim permintaan ke Envoy untuk file. `config_dump` Kebijakan default mencakup setelan berikut:

- Max mencoba ulang — 2
- acara coba lagi gRPC — UNAVAILABLE
- Acara coba lagi HTTP - 503

Note

Tidak mungkin membuat kebijakan coba lagi rute App Mesh yang mencari kode kesalahan HTTP tertentu. Namun, kebijakan coba lagi rute App Mesh dapat mencari `server-error` atau `gateway-error`. Keduanya termasuk 503 kesalahan. Untuk informasi selengkapnya, lihat [Rute](#).

- Acara coba lagi TCP — dan `connect-failure refused-stream`

Note

Tidak mungkin membuat kebijakan coba lagi rute App Mesh yang mencari salah satu dari peristiwa ini. Namun, kebijakan coba lagi rute App Mesh dapat dicari `connection-error`, yang setara `connect-failure` dengan. Untuk informasi selengkapnya, lihat [Rute](#).

- Reset - Utusan mencoba lagi jika server upstream tidak merespons sama sekali (batas waktu).
`disconnect/reset/read`

Pemutus sirkuit default

Saat Anda menerapkan Utusan di App Mesh, nilai default Envoy ditetapkan untuk beberapa pengaturan pemutus sirkuit. Untuk informasi selengkapnya, lihat [cluster. CircuitBreakers.Thresholds dalam dokumentasi](#) Utusan. Pengaturan ini tidak terlihat melalui App Mesh API. Pengaturan hanya terlihat melalui Utusan. Untuk melihat konfigurasi, [aktifkan antarmuka administrasi](#) dan kirim permintaan ke Envoy untuk `file.config_dump`

Jika Anda tidak memiliki jerat di akun Anda sebelum 29 Juli 2020, maka untuk setiap Utusan yang Anda terapkan di mesh yang dibuat pada atau setelah 29 Juli 2020, App Mesh secara efektif menonaktifkan pemutus sirkuit dengan mengubah nilai default Envoy untuk pengaturan yang mengikuti. Jika Anda memiliki jerat di akun Anda sebelum 29 Juli 2020, nilai default Utusan ditetapkan untuk setiap Utusan yang Anda terapkan di App Mesh pada, atau setelah 29 Juli 2020, kecuali [Anda](#) membuka tiket dengan dukungan. AWS Setelah dukungan memproses tiket, maka nilai default App Mesh untuk setelan Envoy berikut ditetapkan oleh App Mesh pada semua Utusan yang Anda terapkan setelah tanggal tiket diproses:

- **max_requests** – 2147483647
- **max_pending_requests** – 2147483647

- **max_connections** – 2147483647
- **max_retries** – 2147483647

Note

Tidak masalah jika Utusan Anda memiliki nilai pemutus sirkuit default Envoy atau App Mesh, Anda tidak dapat mengubah nilainya.

Memperbarui/memigrasi ke Utusan 1.17

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk. AWS App Mesh Setelah 30 September 2026, Anda tidak lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Secret Discovery Service dengan SPIRE

Jika Anda menggunakan SPIRE (SPIFFE Runtime Environment) dengan App Mesh untuk mendistribusikan sertifikat kepercayaan ke layanan Anda, verifikasi bahwa Anda menggunakan setidaknya versi `0.12.0` [agen SPIRE](#) (dirilis Desember 2020). Ini adalah versi pertama yang dapat mendukung versi Utusan setelahnya. `1.16`

Perubahan ekspresi reguler

Mulai dari Utusan `1.17`, App Mesh mengonfigurasi Utusan untuk menggunakan mesin ekspresi [RE2](#) reguler secara default. Perubahan ini terlihat jelas bagi sebagian besar pengguna, tetapi kecocokan di Rute atau Rute Gateway tidak lagi memungkinkan referensi lihat ke depan atau back-reference dalam ekspresi reguler.

Positif dan Negatif melihat ke depan

Positif - Pandangan positif ke depan adalah ekspresi tanda kurung yang dimulai dengan: `?=`

```
(?=example)
```

Ini memiliki utilitas paling banyak saat melakukan penggantian string karena memungkinkan pencocokan string tanpa menggunakan karakter sebagai bagian dari pertandingan. Karena App Mesh tidak mendukung penggantian string regex, kami sarankan Anda menggantinya dengan kecocokan biasa.

```
(example)
```

Negatif - Pandangan negatif ke depan adalah ekspresi tanda kurung yang dimulai dengan. ?!

```
ex(?!amp)le
```

Ekspresi tanda kurung digunakan untuk menegaskan bahwa bagian dari ekspresi tidak cocok dengan input yang diberikan. Dalam kebanyakan kasus, Anda dapat mengganti ini dengan kuantifier nol.

```
ex(amp){0}le
```

Jika ekspresi itu sendiri adalah kelas karakter, Anda dapat meniadakan seluruh kelas dan menandainya opsional menggunakan?.

```
prefix(?![0-9])suffix => prefix[^0-9]?suffix
```

Bergantung pada kasus penggunaan Anda, Anda mungkin juga dapat mengubah rute untuk menangani ini.

```
{
  "routeSpec": {
    "priority": 0,
    "httpRoute": {
      "match": {
        "headers": [
          {
            "name": "x-my-example-header",
            "match": {
              "regex": "^prefix(?!suffix)"
            }
          }
        ]
      }
    }
  }
}
```

```

    ]
  }
}
{
  "routeSpec": {
    "priority": 1,
    "httpRoute": {
      "match": {
        "headers": [
          {
            "name": "x-my-example-header",
            "match": {
              "regex": "^prefix"
            }
          }
        ]
      }
    }
  }
}
}

```

Pencocokan rute pertama mencari header yang dimulai dengan “awalan” tetapi tidak diikuti oleh “akhiran.” Rute kedua bertindak untuk mencocokkan semua header lain yang dimulai dengan “awalan,” termasuk yang diakhiri dengan “akhiran.” Sebaliknya, ini juga dapat dibalik sebagai cara untuk menghilangkan pandangan negatif ke depan.

```

{
  "routeSpec": {
    "priority": 0,
    "httpRoute": {
      "match": {
        "headers": [
          {
            "name": "x-my-example-header",
            "match": {
              "regex": "^prefix.*?suffix"
            }
          }
        ]
      }
    }
  }
}

```

```

    }
  }
}
{
  "routeSpec": {
    "priority": 1,
    "httpRoute": {
      "match": {
        "headers": [
          {
            "name": "x-my-example-header",
            "match": {
              "regex": "^prefix"
            }
          }
        ]
      }
    }
  }
}
}
}

```

Contoh ini membalikkan rute untuk memberikan prioritas yang lebih tinggi ke header yang diakhiri dengan “akhiran”, dan semua header lain yang dimulai dengan “awalan” dicocokkan di rute prioritas rendah.

Referensi belakang

Referensi balik adalah cara untuk menulis ekspresi yang lebih pendek dengan mengulangi ke grup bertanda kurung sebelumnya. Mereka memiliki bentuk ini.

```
(group1)(group2)\1
```

Sebuah garis miring terbalik \ diikuti oleh angka bertindak sebagai pengganti untuk kelompok tanda kurung ke-n dalam ekspresi. Dalam contoh ini, \1 digunakan sebagai cara alternatif untuk (group1) menulis kedua kalinya.

```
(group1)(group2)(group1)
```

Ini dapat dihapus hanya dengan mengganti referensi balik dengan grup yang direferensikan seperti pada contoh.

Agen untuk Utusan

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk. AWS App Mesh Setelah 30 September 2026, Anda tidak lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Agen adalah manajer proses dalam gambar Utusan yang dijual untuk App Mesh. Agen memastikan Utusan tetap berjalan, tetap sehat, dan mengurangi waktu henti. Ini menyaring statistik Utusan dan data tambahan untuk memberikan tampilan suling dari operasi proxy Utusan di App Mesh. Ini dapat membantu Anda memecahkan masalah terkait kesalahan lebih cepat.

Anda dapat menggunakan Agen untuk mengonfigurasi berapa kali Anda ingin memulai ulang proxy Utusan jika proxy menjadi tidak sehat. Jika terjadi kegagalan, Agen mencatat status keluar konklusif saat utusan keluar. Anda dapat menggunakan ini saat memecahkan masalah kegagalan. Agen juga memfasilitasi pengeringan koneksi Envoy, yang membantu membuat aplikasi Anda lebih tahan terhadap kegagalan.

Konfigurasi Agen untuk Utusan menggunakan variabel-variabel ini:

- `APPNET_ENVOY_RESTART_COUNT`— Saat variabel ini disetel ke nilai bukan nol, Agen mencoba memulai ulang proses proxy Utusan hingga nomor yang Anda tetapkan saat dianggap status proses proxy tidak sehat saat polling. Ini membantu mengurangi waktu henti dengan memberikan restart yang lebih cepat dibandingkan dengan tugas atau penggantian pod oleh orkestrator kontainer jika terjadi kegagalan pemeriksaan kesehatan proxy.
- `PID_POLL_INTERVAL_MS`— Saat mengkonfigurasi variabel ini, default disimpan ke 100. Saat disetel ke nilai ini, Anda memungkinkan deteksi lebih cepat dan memulai ulang proses Envoy saat keluar dibandingkan dengan penggantian tugas atau pod melalui pemeriksaan kesehatan orkestrator kontainer.
- `LISTENER_DRAIN_WAIT_TIME_S`— Saat mengkonfigurasi variabel ini, pertimbangkan batas waktu orkestrator kontainer yang ditetapkan untuk menghentikan tugas atau pod. Misalnya, jika nilai ini lebih besar dari batas waktu orkestrator, proxy Envoy hanya dapat menguras durasi hingga orkestrator menghentikan tugas atau pod secara paksa.

- `APPNET_AGENT_ADMIN_MODE`— Ketika variabel ini diatur ke `tcp` atau `uds`, Agen menyediakan antarmuka manajemen lokal. Antarmuka manajemen ini berfungsi sebagai titik akhir yang aman untuk berinteraksi dengan proxy Utusan dan menyediakan yang berikut APIs untuk pemeriksaan kesehatan, data telemetri, dan merangkum kondisi operasi proxy.
 - `GET /status`— Kueri statistik Utusan dan mengembalikan informasi server.
 - `POST /drain_listeners`— Menguras semua pendengar yang masuk.
 - `POST /enableLogging?level=<desired_level>`— Ubah tingkat logging Utusan di semua logger.
 - `GET /stats/prometheus`— Tampilkan statistik Utusan dalam format Prometheus.
 - `GET /stats/prometheus?usedonly`— Hanya tampilkan statistik yang telah diperbarui oleh Utusan.

Untuk informasi selengkapnya tentang variabel konfigurasi Agen, lihat Variabel [konfigurasi Envoy](#).

AWS App Mesh Agen baru disertakan dalam gambar Utusan yang dioptimalkan oleh App Mesh mulai dari versi `1.21.0.0` dan tidak memerlukan alokasi sumber daya tambahan dalam tugas atau pod pelanggan.

Observabilitas App Mesh

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk. AWS App Mesh Setelah 30 September 2026, Anda tidak lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Salah satu manfaat dari bekerja dengan App Mesh adalah visibilitas yang lebih besar ke dalam aplikasi layanan mikro Anda. App Mesh dapat bekerja dengan banyak solusi logging, metrik, dan tracing yang berbeda.

Proxy Envoy dan App Mesh menawarkan jenis alat berikut untuk membantu Anda mendapatkan tampilan yang lebih jelas tentang aplikasi dan proxy Anda:

- [Pencatatan log](#)
- [Metrik-metrik](#)
- [Menelusuri](#)

Pencatatan log

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk. AWS App Mesh Setelah 30 September 2026, Anda tidak lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).


Saat Anda membuat node virtual dan gateway virtual, Anda memiliki opsi untuk mengonfigurasi log akses Envoy. Di konsol, ini ada di bagian Logging dari node virtual dan gateway virtual membuat atau mengedit alur kerja.

Logging

HTTP access logs path - *optional*

The path used to send logging information for the virtual node. App Mesh recommends using the standard out I/O stream.

```
/dev/stdout
```

 Logs must still be ingested by an agent in your application and sent to a destination. This file path only instructs Envoy where to send the logs.

Gambar sebelumnya menunjukkan jalur logging `/dev/stdout` untuk log akses Envoy.

Untuk format, tentukan salah satu dari dua format yang mungkin, `json` atau `text`, dan polanya. `json` mengambil pasangan kunci dan mengubahnya menjadi struct JSON sebelum meneruskannya ke Envoy.

Blok kode berikut menunjukkan representasi JSON yang dapat Anda gunakan di AWS CLI

```
"logging": {
  "accessLog": {
    "file": {
      "path": "/dev/stdout",
      "format" : {
        // Exactly one of json or text should be specified
        "json": [ // json will be implemented with key pairs
          {
            "key": "string",
            "value": "string"
          }
        ]
        "text": "string" //e.g. "%LOCAL_REPLY_BODY%:%RESPONSE_CODE%:path=%REQ(:path)%\n"
      }
    }
  }
}
```

Important

Pastikan untuk memeriksa apakah pola input Anda valid untuk Utusan, atau Utusan akan menolak pembaruan dan menyimpan perubahan terbaru di `error state`

Saat Anda mengirim log akses Envoy ke `/dev/stdout`, log tersebut dicampur dengan log kontainer Envoy. Anda dapat mengekspornya ke penyimpanan log dan layanan pemrosesan seperti CloudWatch Log menggunakan driver log Docker standar seperti `awslogs`. Untuk informasi selengkapnya, lihat [Menggunakan Driver Log `awslogs`](#) di Panduan Pengembang Amazon ECS. Untuk mengekspor hanya log akses Envoy (dan mengabaikan log kontainer Envoy lainnya), Anda dapat menyetel ke `ENVOY_LOG_LEVEL off` Anda dapat mencatat permintaan tanpa string kueri dengan memasukkan string format `%REQ_WITHOUT_QUERY(X?Y):Z%`. Sebagai contoh, lihat [ReqWithoutQueryFormatter](#). Untuk informasi selengkapnya, lihat [Logging akses](#) di dokumentasi Utusan.

Aktifkan log akses di Kubernetes

Saat menggunakan [App Mesh Controller untuk Kubernetes](#), Anda dapat mengonfigurasi node virtual dengan access logging dengan menambahkan konfigurasi logging ke spesifikasi node virtual, seperti yang ditunjukkan pada contoh berikut.

```
---
apiVersion: appmesh.k8s.aws/v1beta2
kind: VirtualNode
metadata:
  name: virtual-node-name
  namespace: namespace
spec:
  listeners:
  - portMapping:
      port: 9080
      protocol: http
  serviceDiscovery:
    dns:
      hostName: hostname
  logging:
    accessLog:
      file:
        path: "/dev/stdout"
```

Cluster Anda harus memiliki log forwarder untuk mengumpulkan log ini, seperti Fluentd. Untuk informasi selengkapnya, lihat, [Siapkan Fluentd sebagai a DaemonSet untuk mengirim log ke CloudWatch Log](#).

Envoy juga menulis berbagai log debugging dari filternya ke `stdout` Log ini berguna untuk mendapatkan wawasan tentang komunikasi Envoy dengan App Mesh dan lalu lintas `service-to-`

service Level logging spesifik Anda dapat dikonfigurasi menggunakan variabel `ENVOY_LOG_LEVEL` lingkungan. Misalnya, teks berikut berasal dari log debug contoh yang menunjukkan klaster yang dicocokkan oleh Envoy untuk permintaan HTTP tertentu.

```
[debug][router] [source/common/router/router.cc:434] [C4][S17419808847192030829]
cluster 'cds_ingress_howto-http2-mesh_color_client_http_8080' match for URL '/ping'
```

Firelens dan Cloudwatch

[Firelens](#) adalah router log kontainer yang dapat Anda gunakan untuk mengumpulkan log untuk Amazon ECS dan. AWS Fargate Anda dapat menemukan contoh penggunaan Firelens di repositori [AWS Sampel](#) kami.

Anda dapat menggunakan CloudWatch untuk mengumpulkan informasi logging serta metrik. Anda dapat menemukan informasi selengkapnya CloudWatch di bagian [Mengeksplor metrik](#) kami di dokumen App Mesh.

Memantau aplikasi Anda menggunakan metrik Envoy

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk. AWS App Mesh Setelah 30 September 2026, Anda tidak lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

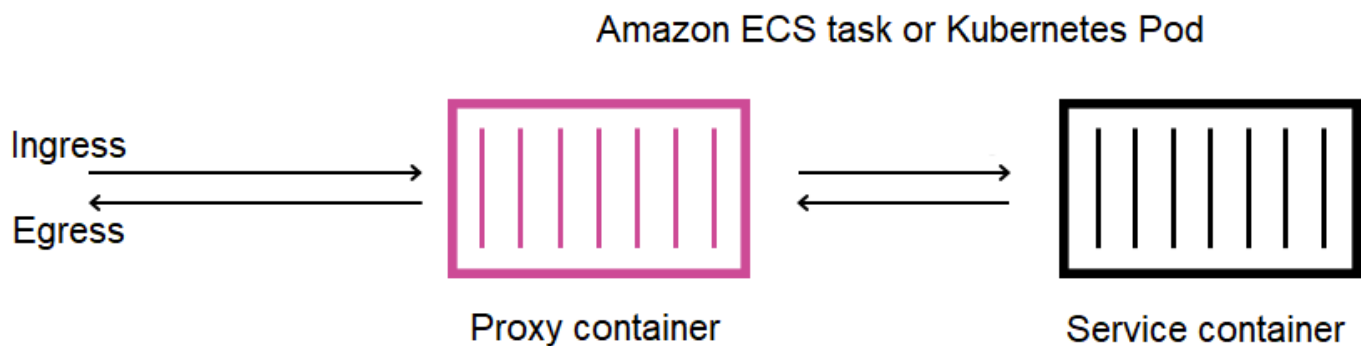
Envoy mengklasifikasikan metriknya ke dalam kategori utama berikut:

- Hilir —Metrik yang berhubungan dengan koneksi dan permintaan yang masuk ke proxy.
- Upstream —Metrik yang berhubungan dengan koneksi keluar dan permintaan yang dibuat oleh proxy.
- Server —Metrik yang menggambarkan keadaan internal utusan. Ini termasuk metrik seperti uptime atau memori yang dialokasikan.

Di App Mesh, proxy mencegat lalu lintas hulu dan hilir. Misalnya, permintaan yang diterima dari klien Anda serta permintaan yang dibuat oleh kontainer layanan Anda diklasifikasikan sebagai lalu lintas hilir oleh Utusan. Untuk membedakan antara berbagai jenis lalu lintas hulu dan hilir ini, App Mesh lebih lanjut mengkategorikan metrik Envoy tergantung pada arah lalu lintas relatif terhadap layanan Anda:

- **Ingress** —Metrik dan sumber daya yang berkaitan dengan koneksi dan permintaan yang mengalir ke wadah layanan Anda.
- **Keluar** —Metrik dan sumber daya yang berkaitan dengan koneksi dan permintaan yang mengalir dari container layanan Anda dan akhirnya keluar dari tugas Amazon ECS atau pod Kubernetes Anda.

Gambar berikut menunjukkan komunikasi antara proxy dan kontainer layanan.



Konvensi penamaan sumber daya

Sangat berguna untuk memahami bagaimana Envoy melihat mesh Anda dan bagaimana sumber dayanya memetakan kembali ke sumber daya yang Anda tentukan di App Mesh. Ini adalah sumber daya Utusan utama yang dikonfigurasi App Mesh:

- **Pendengar** —Alamat dan port yang didengarkan proxy untuk koneksi hilir. Pada gambar sebelumnya, App Mesh membuat pendengar ingress untuk lalu lintas yang masuk ke tugas Amazon ECS atau pod Kubernetes Anda dan pendengar jalan keluar untuk lalu lintas yang meninggalkan container layanan Anda.
- **Cluster** —Grup bernama titik akhir hulu yang dihubungkan oleh proxy dan merutekan lalu lintas ke. Di App Mesh, wadah layanan Anda direpresentasikan sebagai kluster, serta semua node virtual lain yang dapat disambungkan oleh layanan Anda.

- Rute —Ini sesuai dengan rute yang Anda tentukan di mesh Anda. Mereka berisi kondisi dimana proxy cocok dengan permintaan serta kluster target yang dikirimkan permintaan.
- Endpoint dan tugas beban cluster —Alamat IP cluster hulu. AWS Cloud Map Saat menggunakan mekanisme penemuan layanan Anda untuk node virtual, App Mesh mengirimkan instance layanan yang ditemukan sebagai sumber daya titik akhir ke proxy Anda.
- Rahasia —Ini termasuk, tetapi tidak terbatas pada, kunci enkripsi dan sertifikat TLS Anda. Saat menggunakan AWS Certificate Manager sebagai sumber untuk sertifikat klien dan server, App Mesh mengirimkan sertifikat publik dan pribadi ke proxy Anda sebagai sumber daya rahasia.

App Mesh menggunakan skema yang konsisten untuk memberi nama sumber daya Envoy yang dapat Anda gunakan untuk menghubungkan kembali ke mesh Anda.

Memahami skema penamaan untuk pendengar dan cluster penting dalam memahami metrik Envoy di App Mesh.

Nama pendengar

Pendengar diberi nama menggunakan format berikut:

```
lds_<traffic direction>_<listener IP address>_<listening port>
```

Anda biasanya akan melihat pendengar berikut dikonfigurasi di Utusan:

- lds_ingress_0.0.0.0_15000
- lds_egress_0.0.0.0_15001

Menggunakan plugin Kubernetes CNI atau aturan tabel IP, lalu lintas di tugas Amazon ECS atau pod Kubernetes diarahkan ke port dan. 15000 15001 App Mesh mengonfigurasi Envoy dengan dua pendengar ini untuk menerima lalu lintas masuk (masuk) dan keluar (keluar). Jika Anda tidak memiliki listener yang dikonfigurasi pada node virtual Anda, Anda seharusnya tidak melihat pendengar ingress.

Nama cluster

Sebagian besar cluster menggunakan format berikut:

```
cds_<traffic direction>_<mesh name>_<virtual node name>_<protocol>_<port>
```

Node virtual yang berkomunikasi dengan layanan Anda masing-masing memiliki kluster mereka sendiri. Seperti disebutkan sebelumnya, App Mesh membuat cluster untuk layanan yang berjalan di sebelah Envoy sehingga proxy dapat mengirim lalu lintas masuk ke sana.

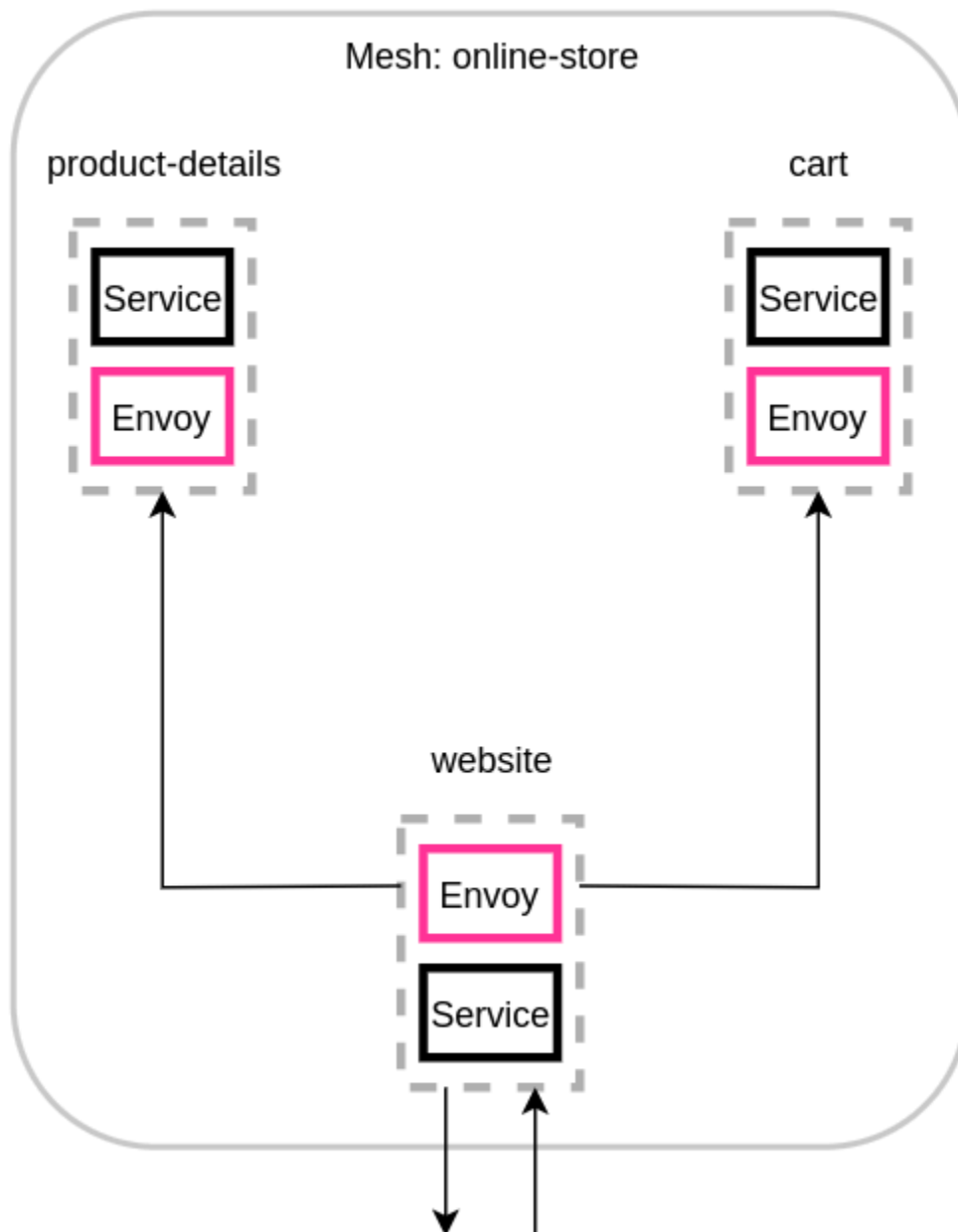
Misalnya, jika Anda memiliki node virtual bernama `my-virtual-node` yang mendengarkan lalu lintas http pada port `8080` dan node virtual tersebut berada dalam mesh bernama `my-mesh`, App Mesh membuat kluster bernama `cds_ingress_my-mesh_my-virtual-node_http_8080`. Cluster ini berfungsi sebagai tujuan untuk lalu lintas ke dalam `my-virtual-node` wadah layanan.

App Mesh juga dapat membuat jenis cluster khusus tambahan berikut. Cluster lain ini tidak selalu sesuai dengan sumber daya yang Anda definisikan secara eksplisit di mesh Anda.

- Cluster digunakan untuk menjangkau AWS layanan lain. Jenis ini memungkinkan mesh Anda menjangkau sebagian besar AWS layanan secara default: `cds_egress_<mesh name>_amazonaws`.
- Cluster digunakan untuk melakukan routing untuk gateway virtual. Ini umumnya dapat diabaikan dengan aman:
 - Untuk pendengar tunggal: `cds_ingress_<mesh name>_<virtual gateway name>_self_redirect_<protocol>_<port>`
 - Untuk beberapa pendengar: `cds_ingress_<mesh name>_<virtual gateway name>_self_redirect_<ingress_listener_port>_<protocol>_<port>`
- Cluster yang merupakan titik akhir yang dapat Anda tentukan, seperti TLS, saat Anda mengambil rahasia menggunakan Envoy's Secret Discovery Service: `static_cluster_sds_unix_socket`

Contoh metrik aplikasi

Untuk mengilustrasikan metrik yang tersedia di Envoy, contoh aplikasi berikut memiliki tiga node virtual. Layanan virtual, router virtual, dan rute di mesh dapat diabaikan karena tidak tercermin dalam metrik Envoy. Dalam contoh ini, semua layanan mendengarkan lalu lintas http pada port `8080`.



Sebaiknya tambahkan variabel lingkungan `ENABLE_ENVOY_STATS_TAGS=1` ke container proxy Envoy yang berjalan di mesh Anda. Ini menambahkan dimensi metrik berikut ke semua metrik yang dipancarkan oleh proxy:

- `appmesh.mesh`
- `appmesh.virtual_node`
- `appmesh.virtual_gateway`

Tag ini diatur ke nama mesh, node virtual, atau gateway virtual untuk memungkinkan metrik pemfilteran menggunakan nama sumber daya di mesh Anda.

Nama sumber daya

Proxy node virtual situs web memiliki sumber daya berikut:

- Dua pendengar untuk lalu lintas masuk dan keluar:
 - `lds_ingress_0.0.0.0_15000`
 - `lds_egress_0.0.0.0_15001`
- Dua cluster jalan keluar, mewakili dua ujung belakang simpul virtual:
 - `cds_egress_online-store-product-details_http_8080`
 - `cds_egress_online-store-cart_http_8080`
- Kluster ingress untuk wadah layanan situs web:
 - `cds_ingress_online-store-website_http_8080`

Contoh metrik pendengar

- `listener.0.0.0.0_15000.downstream_cx_active`—Jumlah koneksi jaringan masuk aktif ke Utusan.
- `listener.0.0.0.0_15001.downstream_cx_active`—Jumlah koneksi jaringan keluar aktif ke Utusan. Koneksi yang dibuat oleh aplikasi Anda ke layanan eksternal termasuk dalam hitungan ini.
- `listener.0.0.0.0_15000.downstream_cx_total`—Jumlah total koneksi jaringan ingress ke Envoy.
- `listener.0.0.0.0_15001.downstream_cx_total`—Jumlah total koneksi jaringan keluar ke Utusan.

Untuk set lengkap metrik pendengar, lihat [Statistik](#) dalam dokumentasi Utusan.

Contoh metrik cluster

- `cluster_manager.active_clusters`—Jumlah total cluster yang telah dibuat oleh Utusan setidaknya satu koneksi.
- `cluster_manager.warming_clusters`—Jumlah total cluster yang belum terhubung dengan Utusan.

Metrik cluster berikut menggunakan format `cluster.<cluster name>.<metric name>`. Nama-nama metrik ini unik untuk contoh aplikasi dan dipancarkan oleh wadah Utusan situs web:

- `cluster.cds_egress_online-store_product-details_http_8080.upstream_cx_total`— Jumlah total koneksi antara situs web dan detail produk.
- `cluster.cds_egress_online-store_product-details_http_8080.upstream_cx_connect_fail`— Jumlah total koneksi yang gagal antara situs web dan detail produk.
- `cluster.cds_egress_online-store_product-details_http_8080.health_check.failure`— Jumlah total pemeriksaan kesehatan yang gagal antara situs web dan detail produk.
- `cluster.cds_egress_online-store_product-details_http_8080.upstream_rq_total`— Jumlah total permintaan yang dibuat antara situs web dan detail produk.
- `cluster.cds_egress_online-store_product-details_http_8080.upstream_rq_time`— Waktu yang dibutuhkan oleh permintaan yang dibuat antara situs web dan detail produk.
- `cluster.cds_egress_online-store_product-details_http_8080.upstream_rq_2xx`— Jumlah tanggapan HTTP 2xx yang diterima oleh situs web dari detail produk.

Untuk set lengkap metrik HTTP, lihat [Statistik dalam dokumentasi](#) Utusan.

Metrik server manajemen

Envoy juga memancarkan metrik yang terkait dengan koneksinya ke bidang kontrol App Mesh, yang bertindak sebagai server manajemen Envoy. Kami menyarankan untuk memantau beberapa metrik ini sebagai cara untuk memberi tahu Anda ketika proxy Anda menjadi tidak sinkron dari bidang kontrol untuk waktu yang lama. Hilangnya konektivitas ke bidang kontrol atau pembaruan yang gagal mencegah proxy Anda menerima konfigurasi baru dari App Mesh, termasuk perubahan mesh yang dilakukan melalui App Mesh. APIs

- `control_plane.connected_state`—Metrik ini diatur ke 1 saat proxy terhubung ke App Mesh, jika tidak maka 0.

- `*.update_rejected`—Jumlah total pembaruan konfigurasi yang ditolak oleh Utusan. Ini biasanya karena kesalahan konfigurasi pengguna. Misalnya, jika Anda mengonfigurasi App Mesh untuk membaca sertifikat TLS dari file yang tidak dapat dibaca oleh Envoy, pembaruan yang berisi jalur ke sertifikat tersebut akan ditolak.
 - Untuk Listener diperbarui ditolak, statistiknya akan `listener_manager.lds.update_rejected`
 - Untuk Cluster yang diperbarui ditolak, statistiknya akan `cluster_manager.cds.update_rejected`.
- `*.update_success`—Jumlah pembaruan konfigurasi yang berhasil dibuat oleh App Mesh ke proxy Anda. Ini termasuk payload konfigurasi awal yang dikirim saat container Envoy baru dimulai.
 - Untuk keberhasilan yang diperbarui Listener, statistiknya akan `listener_manager.lds.update_success`
 - Untuk keberhasilan yang diperbarui Cluster, statistiknya akan `cluster_manager.cds.update_success`.

Untuk kumpulan metrik server manajemen, lihat [Server Manajemen dalam dokumentasi](#) Utusan.

Mengekspor metrik

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk AWS App Mesh. Setelah 30 September 2026, Anda tidak lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Utusan memancarkan banyak statistik pada operasinya sendiri dan berbagai dimensi pada lalu lintas masuk dan keluar. Untuk mempelajari lebih lanjut tentang statistik Utusan, lihat [Statistik dalam dokumentasi](#) Utusan. Metrik ini tersedia melalui `/stats` titik akhir pada port administrasi proxy, yang biasanya 9901

statAwalan akan berbeda tergantung pada apakah Anda menggunakan pendengar tunggal atau beberapa. Di bawah ini adalah beberapa contoh untuk menggambarkan perbedaannya.

⚠ Warning

Jika Anda memperbarui listener tunggal Anda ke fitur multiple listener, Anda dapat menghadapi perubahan yang melanggar karena awalan stat diperbarui yang diilustrasikan dalam tabel berikut.

Kami sarankan Anda menggunakan gambar Utusan 1.22.2.1-prod atau yang lebih baru. Ini memungkinkan Anda untuk melihat nama metrik yang serupa di titik akhir Prometheus Anda.

Single Listener (SL) /Statistik yang ada dengan awalan pendengar "ingress"	Beberapa Pendengar (ML) / Statistik baru dengan "ingress.<protocol>. <port>"awalan pendengar	
<pre>http.*ingress*.rds .rds_ingress_http_ 5555.version_text</pre>	<pre>http.*ingress.http .5555*.rds.rds_ing ress_http_5555.ver sion_text http.*ingress.http .6666*.rds.rds_ing ress_http_6666.ver sion_text</pre>	
<pre>listener.0.0.0.0_1 5000.http.*ingress *.downstream_rq_2xx</pre>	<pre>listener.0.0.0.0_1 5000.http.*ingress .http.5555*.downst ream_rq_2xx listener.0.0.0.0_1 5000.http.*ingress .http.6666*.downst ream_rq_2xx</pre>	

Single Listener (SL) / Statistik yang ada dengan awalan pendengar "ingress"	Beberapa Pendengar (ML) / Statistik baru dengan "ingress. <protocol>. <port>"awalan pendengar
<code>http.*ingress*.downstream_cx_length_ms</code>	<code>http.*ingress.http.5555*.downstream_cx_length_ms</code> <code>http.*ingress.http.6666*.downstream_cx_length_ms</code>

Untuk informasi selengkapnya tentang titik akhir statistik, lihat [Titik akhir Statistik](#) dalam dokumentasi Utusan. Untuk informasi selengkapnya tentang antarmuka administrasi, lihat [Aktifkan antarmuka administrasi proxy Envoy](#).

Prometheus untuk App Mesh dengan Amazon EKS

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk AWS App Mesh. Setelah 30 September 2026, Anda tidak lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Prometheus adalah toolkit pemantauan dan peringatan sumber terbuka. Salah satu kemampuannya adalah menentukan format untuk memancarkan metrik yang dapat dikonsumsi oleh sistem lain. Untuk informasi lebih lanjut tentang Prometheus, lihat [Ikhtisar](#) dalam dokumentasi Prometheus. Utusan dapat memancarkan metriknya melalui titik akhir statistiknya dengan meneruskan parameter `/stats?format=prometheus`

Untuk pelanggan yang menggunakan Envoy image build v1.22.2.1-prod, ada dua dimensi tambahan untuk menunjukkan statistik spesifik pendengar ingress:

- `appmesh.listener_protocol`

- `appmesh.listener_port`

Di bawah ini adalah perbandingan antara statistik Prometheus yang ada vs statistik baru.

- Statistik yang ada dengan awalan pendengar “ingress”

```
envoy_http_downstream_rq_xx{appmesh_mesh="multiple-listeners-mesh",appmesh_virtual_node="foodteller-vn",envoy_response_code_class="2",envoy_http_conn_manager_prefix="ingress"} 931433
```

- Statistik baru dengan “ingress. <protocol>. <port>”+ Appmesh Envoy Image v1.22.2.1-prod atau yang lebih baru

```
envoy_http_downstream_rq_xx{appmesh_mesh="multiple-listeners-mesh",appmesh_virtual_node="foodteller-vn",envoy_response_code_class="2",appmesh_listener_protocol="http",appmesh_listener_port="5555"} 20
```

- Statistik baru dengan “ingress. <protocol>. <port>”+ Utusan kustom Imagebuild

```
envoy_http_http_5555_downstream_rq_xx{appmesh_mesh="multiple-listeners-mesh",appmesh_virtual_node="foodteller-vn",envoy_response_code_class="2",envoy_http_conn_manager_prefix="ingress"} 15983
```

Untuk beberapa pendengar, `cluster cds_ingress_<mesh name>_<virtual gateway name>_self_redirect_<ingress_listener_port>_<protocol>_<port>` khusus akan menjadi pendengar khusus.

- Statistik yang ada dengan awalan pendengar “ingress”

```
envoy_cluster_assignment_stale{appmesh_mesh="multiple-listeners-mesh",appmesh_virtual_gateway="telligateway-vg",Mesh="multiple-listeners-mesh",VirtualGateway="telligateway-vg",envoy_cluster_name="cds_ingress_multiple-listeners-mesh_telligateway-vg_self_redirect_http_15001"} 0
```

- Statistik baru dengan “ingress. <protocol>. <port>”

```
envoy_cluster_assignment_stale{appmesh_mesh="multiple-listeners-mesh",appmesh_virtual_gateway="telligateway-
```

```
vg",envoy_cluster_name="cds_ingress_multiple-listeners-mesh_tellergateway-  
vg_self_redirect_1111_http_15001"} 0  
envoy_cluster_assignment_stale{appmesh_mesh="multiple-  
listeners-mesh",appmesh_virtual_gateway="tellergateway-  
vg",envoy_cluster_name="cds_ingress_multiple-listeners-mesh_tellergateway-  
vg_self_redirect_2222_http_15001"} 0
```

Instalasi Prometheus

1. Tambahkan repositori EKS ke Helm:

```
helm repo add eks https://aws.github.io/eks-charts
```

2. Instal App Mesh Prometheus

```
helm upgrade -i appmesh-prometheus eks/appmesh-prometheus \  
--namespace appmesh-system
```

Contoh Prometheus

Berikut ini adalah contoh pembuatan penyimpanan persisten PersistentVolumeClaim untuk Prometheus.

```
helm upgrade -i appmesh-prometheus eks/appmesh-prometheus \  
--namespace appmesh-system \  
--set retention=12h \  
--set persistentVolumeClaim.claimName=prometheus
```

Panduan untuk menggunakan Prometheus

- [App Mesh dengan Eks — Observabilitas: Prometheus](#)

Untuk mempelajari lebih lanjut tentang Prometheus dan Prometheus dengan Amazon EKS

- [Dokumentasi Prometheus](#)

- EKS - [Kontrol metrik bidang dengan Prometheus](#)

CloudWatch untuk App Mesh

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk AWS App Mesh. Setelah 30 September 2026, Anda tidak lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Memancarkan statistik Utusan ke dari Amazon EKS CloudWatch

Anda dapat menginstal CloudWatch Agen ke kluster Anda dan mengonfigurasinya untuk mengumpulkan subset metrik dari proxy Anda. Jika Anda belum memiliki cluster Amazon EKS, maka Anda dapat membuatnya dengan langkah-langkah di [Walkthrough: App Mesh dengan Amazon EKS aktif](#). GitHub Anda dapat menginstal aplikasi sampel ke cluster dengan mengikuti panduan yang sama.

Untuk mengatur izin IAM yang sesuai untuk kluster Anda dan menginstal agen, ikuti langkah-langkah di [Instal Agen dengan Koleksi Metrik CloudWatch Prometheus](#). Instalasi default berisi konfigurasi scrape Prometheus yang menarik subset statistik Utusan yang berguna. Untuk informasi selengkapnya, lihat [Metrik Prometheus](#) untuk App Mesh.

Untuk membuat CloudWatch dasbor kustom App Mesh yang dikonfigurasi untuk menampilkan metrik yang dikumpulkan agen, ikuti langkah-langkah dalam tutorial [Melihat Metrik Prometheus Anda](#). Grafik Anda akan mulai terisi dengan metrik yang sesuai saat lalu lintas memasuki aplikasi App Mesh.

Metrik penyaringan untuk CloudWatch

[Ekstensi metrik](#) App Mesh menyediakan subset metrik berguna yang memberi Anda wawasan tentang perilaku sumber daya yang Anda tentukan di mesh Anda. Karena CloudWatch agen mendukung pengikisan metrik Prometheus, Anda dapat memberikan konfigurasi scrape untuk memilih metrik yang ingin Anda tarik dari Utusan dan kirim ke CloudWatch

[Anda dapat menemukan contoh metrik pengikisan menggunakan Prometheus di panduan Ekstensi Metrik kami.](#)

CloudWatch Contoh

Anda dapat menemukan konfigurasi sampel CloudWatch di [repositori AWS Sampel](#) kami.

Walkthrough untuk menggunakan CloudWatch

- [Tambahkan kemampuan pemantauan dan pencatatan](#) di [bengkel App Mesh](#) kami.
- [App Mesh dengan EKS — Observabilitas: CloudWatch](#)
- [Menggunakan ekstensi metrik App Mesh di ECS](#)

Ekstensi metrik untuk App Mesh

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk. AWS App Mesh Setelah 30 September 2026, Anda tidak lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Utusan menghasilkan ratusan metrik yang dipecah menjadi beberapa dimensi yang berbeda. Metrik tidak mudah dalam cara mereka berhubungan kembali ke App Mesh. Dalam kasus layanan virtual, tidak ada mekanisme untuk mengetahui dengan pasti layanan virtual mana yang berkomunikasi ke node virtual atau gateway virtual tertentu.

Ekstensi metrik App Mesh meningkatkan proxy Envoy yang berjalan di mesh Anda. Peningkatan ini memungkinkan proxy untuk memancarkan metrik tambahan yang mengetahui sumber daya yang Anda tentukan. Subset kecil metrik tambahan ini akan membantu memberi Anda wawasan yang lebih besar tentang perilaku sumber daya yang Anda tentukan di App Mesh.

Untuk mengaktifkan ekstensi metrik App Mesh, setel variabel lingkungan `APPMESH_METRIC_EXTENSION_VERSION` ke 1.

```
APPMESH_METRIC_EXTENSION_VERSION=1
```

Untuk informasi selengkapnya tentang variabel konfigurasi Envoy, lihat. [Variabel konfigurasi utusan](#)

Metrik Terkait Lalu Lintas Masuk

- **ActiveConnectionCount**

- `envoy.appmesh.ActiveConnectionCount`— Jumlah koneksi TCP aktif.
- Dimensi - Mesh, VirtualNode, VirtualGateway

- **NewConnectionCount**

- `envoy.appmesh.NewConnectionCount`— Jumlah total koneksi TCP.
- Dimensi - Mesh, VirtualNode, VirtualGateway

- **ProcessedBytes**

- `envoy.appmesh.ProcessedBytes`— Total byte TCP yang dikirim ke dan diterima dari klien hilir.
- Dimensi - Mesh, VirtualNode, VirtualGateway

- **RequestCount**

- `envoy.appmesh.RequestCount`— Jumlah permintaan HTTP yang diproses.
- Dimensi - Mesh, VirtualNode, VirtualGateway

- **GrpcRequestCount**

- `envoy.appmesh.GrpcRequestCount`— Jumlah permintaan GPRC yang diproses.
- Dimensi - Mesh, VirtualNode, VirtualGateway

Metrik Terkait Lalu Lintas Keluar

Anda akan melihat dimensi yang berbeda pada metrik keluar Anda berdasarkan apakah mereka berasal dari node virtual atau gateway virtual.

- **TargetProcessedBytes**

- `envoy.appmesh.TargetProcessedBytes`— Total byte TCP yang dikirim ke dan diterima dari target hulu Utusan.
- Dimensi:
 - Dimensi simpul virtual - Mesh, VirtualNode, TargetVirtualService, TargetVirtualNode
 - Dimensi gateway virtual - Mesh, VirtualGateway, TargetVirtualService, TargetVirtualNode

- **HTTPCode_Target_2XX_Count**

- `envoy.appmesh.HTTPCode_Target_2XX_Count`— Jumlah permintaan HTTP ke target hulu Utusan yang menghasilkan respons HTTP 2xx.

- Dimensi:
 - Dimensi simpul virtual - Mesh, VirtualNode, TargetVirtualService, TargetVirtualNode
 - Dimensi gateway virtual - Mesh, VirtualGateway, TargetVirtualService, TargetVirtualNode
- **HTTPCode_Target_3XX_Count**
 - `envoy.appmesh.HTTPCode_Target_3XX_Count`— Jumlah permintaan HTTP ke target hulu Utusan yang menghasilkan respons HTTP 3xx.
 - Dimensi:
 - Dimensi simpul virtual - Mesh, VirtualNode, TargetVirtualService, TargetVirtualNode
 - Dimensi gateway virtual - Mesh, VirtualGateway, TargetVirtualService, TargetVirtualNode
- **HTTPCode_Target_4XX_Count**
 - `envoy.appmesh.HTTPCode_Target_4XX_Count`— Jumlah permintaan HTTP ke target hulu Utusan yang menghasilkan respons HTTP 4xx.
 - Dimensi:
 - Dimensi simpul virtual - Mesh, VirtualNode, TargetVirtualService, TargetVirtualNode
 - Dimensi gateway virtual - Mesh, VirtualGateway, TargetVirtualService, TargetVirtualNode
- **HTTPCode_Target_5XX_Count**
 - `envoy.appmesh.HTTPCode_Target_5XX_Count`— Jumlah permintaan HTTP ke target hulu Utusan yang menghasilkan respons HTTP 5xx.
 - Dimensi:
 - Dimensi simpul virtual - Mesh, VirtualNode, TargetVirtualService, TargetVirtualNode
 - Dimensi gateway virtual - Mesh, VirtualGateway, TargetVirtualService, TargetVirtualNode
- **RequestCountPerTarget**
 - `envoy.appmesh.RequestCountPerTarget`— Jumlah permintaan yang dikirim ke target hulu Utusan.
 - Dimensi:
 - Dimensi simpul virtual - Mesh, VirtualNode, TargetVirtualService, TargetVirtualNode
 - Dimensi gateway virtual - Mesh, VirtualGateway, TargetVirtualService, TargetVirtualNode
- **TargetResponseTime**
 - `envoy.appmesh.TargetResponseTime`— Waktu berlalu dari saat permintaan dibuat ke target hulu Utusan hingga saat respons penuh diterima.

- Dimensi simpul virtual - Mesh, VirtualNode, TargetVirtualService, TargetVirtualNode
- Dimensi gateway virtual - Mesh, VirtualGateway, TargetVirtualService, TargetVirtualNode

Datadog untuk App Mesh

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk. AWS App Mesh Setelah 30 September 2026, Anda tidak lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Datadog adalah aplikasi pemantauan dan keamanan untuk pemantauan ujung ke ujung, metrik, dan pencatatan aplikasi cloud. Datadog membuat infrastruktur, aplikasi, dan aplikasi pihak ketiga Anda sepenuhnya dapat diamati.

Instalasi Datadog

- EKS - Untuk mengatur Datadog dengan EKS, ikuti langkah-langkah ini dari dokumen [Datadog](#).
- [ECS EC2 - Untuk mengatur Datadog dengan ECS EC2, ikuti langkah-langkah ini dari dokumen Datadog](#).

Untuk mempelajari lebih lanjut tentang Datadog

- [Dokumentasi Datadog](#)

Pelacakan

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk. AWS App Mesh Setelah 30 September 2026, Anda tidak lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi

lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Important

Untuk menerapkan penelusuran sepenuhnya, Anda harus memperbarui aplikasi Anda. Untuk melihat semua data yang tersedia dari layanan yang Anda pilih, Anda harus menginstrumentasikan aplikasi Anda menggunakan pustaka yang berlaku.

Monitor App Mesh dengan AWS X-Ray

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk AWS App Mesh. Setelah 30 September 2026, Anda tidak lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

AWS X-Ray adalah layanan yang menyediakan alat yang memungkinkan Anda melihat, memfilter, dan mendapatkan wawasan tentang data yang dikumpulkan dari permintaan yang disajikan aplikasi Anda. Wawasan ini membantu Anda mengidentifikasi masalah dan peluang untuk mengoptimalkan aplikasi Anda. Anda dapat melihat informasi terperinci tentang permintaan dan tanggapan, dan panggilan hilir yang dilakukan aplikasi Anda ke AWS layanan lain.

X-Ray terintegrasi dengan App Mesh untuk mengelola layanan mikro Envoy Anda. Data jejak dari Utusan dikirim ke daemon X-Ray yang berjalan di wadah Anda.

Terapkan X-Ray dalam kode aplikasi Anda menggunakan panduan [SDK](#) khusus untuk bahasa Anda.

Aktifkan penelusuran X-Ray melalui App Mesh

- Tergantung pada jenis layanan:
 - ECS - Dalam definisi wadah proxy Envoy, atur variabel `ENABLE_ENVOY_XRAY_TRACING` lingkungan ke 1 dan variabel `XRAY_DAEMON_PORT` lingkungan ke. `2000`

- EKS - Dalam konfigurasi App Mesh Controller, sertakan `--set tracing.enabled=true` dan `--set tracing.provider=x-ray`.
- Dalam wadah X-Ray Anda, paparkan port 2000 dan jalankan sebagai pengguna1337.

Contoh X-Ray

Definisi wadah Utusan untuk Amazon ECS

```
{
  "name": "envoy",
  "image": "840364872350.dkr.ecr.us-west-2.amazonaws.com/aws-appmesh-
envoy:v1.15.1.0-prod",
  "essential": true,
  "environment": [
    {
      "name": "APPMESH_VIRTUAL_NODE_NAME",
      "value": "mesh/myMesh/virtualNode/myNode"
    },
    {
      "name": "ENABLE_ENVOY_XRAY_TRACING",
      "value": "1"
    }
  ],
  "healthCheck": {
    "command": [
      "CMD-SHELL",
      "curl -s http://localhost:9901/server_info | cut -d' ' -f3 | grep -q live"
    ],
    "startPeriod": 10,
    "interval": 5,
    "timeout": 2,
    "retries": 3
  }
}
```

Memperbarui pengontrol App Mesh untuk Amazon EKS

```
helm upgrade -i appmesh-controller eks/appmesh-controller \
--namespace appmesh-system \
```

```
--set region=${AWS_REGION} \  
--set serviceAccount.create=false \  
--set serviceAccount.name=appmesh-controller \  
--set tracing.enabled=true \  
--set tracing.provider=x-ray
```

Panduan untuk menggunakan X-Ray

- [Monitor dengan AWS X-Ray](#)
- [App Mesh dengan Amazon EKS - Observabilitas: X-Ray](#)
- [Penelusuran terdistribusi dengan X-Ray di Workshop AWS App Mesh](#)

Untuk mempelajari lebih lanjut tentang AWS X-Ray

- [AWS Dokumentasi X-Ray](#)

Memecahkan Masalah AWS X-Ray dengan App Mesh

- [Tidak dapat melihat jejak AWS X-Ray untuk aplikasi saya.](#)

Jaeger untuk App Mesh dengan Amazon EKS

Jaeger adalah sistem penelusuran terdistribusi open source, ujung ke ujung. Ini dapat digunakan untuk profil jaringan dan untuk pemantauan. Jaeger juga dapat membantu Anda memecahkan masalah aplikasi cloud native yang kompleks.

[Untuk menerapkan Jaeger ke dalam kode aplikasi Anda, Anda dapat menemukan panduan khusus untuk bahasa Anda di pustaka penelusuran dokumentasi Jaeger.](#)

Menginstal Jaeger menggunakan Helm

1. Tambahkan repositori EKS ke Helm:

```
helm repo add eks https://aws.github.io/eks-charts
```

2. Instal App Mesh Jaeger

```
helm upgrade -i appmesh-jaeger eks/appmesh-jaeger \  
--namespace appmesh-system
```

Contoh Jaeger

Berikut ini adalah contoh pembuatan penyimpanan persisten PersistentVolumeClaim untuk Jaeger.

```
helm upgrade -i appmesh-controller eks/appmesh-controller \  
--namespace appmesh-system \  
--set tracing.enabled=true \  
--set tracing.provider=jaeger \  
--set tracing.address=appmesh-jaeger.appmesh-system \  
--set tracing.port=9411
```

Panduan untuk menggunakan Jaeger

- [App Mesh dengan Eks — Observabilitas: Jaeger](#)

Untuk mempelajari lebih lanjut tentang Jaeger

- [Dokumentasi Jaeger](#)

Datadog untuk melacak

Datadog dapat digunakan untuk melacak serta metrik. Untuk informasi lebih lanjut dan petunjuk instalasi, temukan panduan khusus untuk bahasa aplikasi Anda dalam dokumentasi [Datadog](#).

Perkakas App Mesh

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk AWS App Mesh. Setelah 30 September 2026, Anda tidak lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

App Mesh memberi pelanggan kemampuan untuk berinteraksi dengannya secara APIs tidak langsung menggunakan alat-alat seperti:

- CloudFormation
- AWS Cloud Development Kit (AWS CDK)
- App Mesh Controller untuk Kubernetes
- Terraform

App Mesh dan CloudFormation

CloudFormation adalah layanan yang memungkinkan Anda membuat template dengan semua sumber daya yang Anda butuhkan untuk aplikasi Anda, dan kemudian CloudFormation akan mengkonfigurasi dan menyediakan sumber daya untuk Anda. Ini juga akan mengkonfigurasi semua dependensi, sehingga Anda dapat lebih fokus pada aplikasi Anda dan lebih sedikit mengelola sumber daya.

Untuk informasi selengkapnya dan contoh tentang penggunaan CloudFormation dengan App Mesh, lihat [CloudFormation dokumentasi](#).

App Mesh dan AWS CDK

AWS CDK adalah kerangka kerja pengembangan untuk menggunakan kode untuk menentukan infrastruktur cloud Anda dan menggunakannya CloudFormation untuk menyediakannya. AWS CDK mendukung beberapa bahasa pemrograman termasuk TypeScript, Python JavaScript, Java, dan C #/.Net.

Untuk informasi selengkapnya tentang penggunaan AWS CDK dengan App Mesh, lihat [AWS CDK dokumentasi](#).

Pengontrol App Mesh untuk Kubernetes

Pengontrol App Mesh untuk Kubernetes membantu Anda mengelola resource App Mesh untuk kluster Kubernetes dan menyuntikkan sidecar ke dalam pod. Pengontrol ini khusus untuk digunakan dengan Amazon EKS dan memungkinkan Anda mengelola sumber daya Anda dengan cara yang asli dari Kubernetes.

Untuk informasi selengkapnya tentang pengontrol App Mesh, lihat [dokumentasi App Mesh Controller](#).

App Mesh dan Terraform

[Terraform](#) adalah infrastruktur sumber terbuka sebagai alat perangkat lunak kode. Terraform dapat mengelola layanan cloud menggunakan CLI mereka dan berinteraksi APIs dengan menggunakan file konfigurasi deklaratif.

Untuk melihat lebih lanjut tentang menggunakan App Mesh dengan Terraform, lihat dokumentasi [Terraform](#).

Bekerja dengan jerat bersama

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk AWS App Mesh. Setelah 30 September 2026, Anda tidak lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Anda dapat membagikan mesh App Mesh di seluruh AWS akun menggunakan AWS Resource Access Manager layanan. Mesh bersama memungkinkan sumber daya yang dibuat oleh AWS akun yang berbeda untuk berkomunikasi satu sama lain dalam mesh yang sama.

AWS Akun dapat berupa pemilik sumber daya mesh, konsumen mesh, atau keduanya. Konsumen dapat membuat sumber daya dalam mesh yang dibagikan dengan akun mereka. Pemilik dapat membuat sumber daya di mesh apa pun yang dimiliki akun. Pemilik mesh dapat berbagi mesh dengan jenis konsumen mesh berikut.

- AWS Akun tertentu di dalam atau di luar organisasinya di AWS Organizations
- Unit organisasi di dalam organisasinya di AWS Organizations
- Seluruh organisasinya di AWS Organizations

Untuk end-to-end berjalan-jalan berbagi mesh, lihat [Cross-account mesh walk through](#) on GitHub.

Memberikan izin untuk berbagi jerat

Saat berbagi jerat di seluruh akun, ada izin yang diperlukan untuk prinsipal IAM yang berbagi mesh dan izin tingkat sumber daya yang diperlukan untuk mesh itu sendiri.

Memberikan izin untuk berbagi mesh

Satu set izin minimum diperlukan untuk prinsipal IAM untuk berbagi mesh. Sebaiknya gunakan kebijakan IAM yang `AWSResourceAccessManagerFullAccess` dikelola `AWSAppMeshFullAccess` dan dikelola untuk memastikan kepala sekolah IAM Anda memiliki izin yang diperlukan untuk berbagi dan menggunakan mesh bersama.

Jika Anda menggunakan kebijakan IAM kustom, tindakan `appmesh:PutMeshPolicy`, `appmesh:GetMeshPolicy`, dan `appmesh>DeleteMeshPolicy` tindakan diperlukan. Ini adalah tindakan IAM khusus izin. Jika prinsipal IAM tidak memiliki izin ini diberikan, kesalahan akan terjadi saat mencoba membagikan mesh menggunakan layanan. AWS RAM

Untuk informasi selengkapnya tentang cara AWS Resource Access Manager layanan menggunakan IAM, lihat [Cara AWS RAM menggunakan IAM](#) di AWS Resource Access Manager Panduan Pengguna.

Memberikan izin untuk mesh

Mesh bersama memiliki izin berikut.

- Konsumen dapat membuat daftar dan mendeskripsikan semua sumber daya dalam mesh yang dibagikan dengan akun.
- Pemilik dapat membuat daftar dan mendeskripsikan semua sumber daya di mesh mana pun yang dimiliki akun.
- Pemilik dan konsumen dapat memodifikasi sumber daya dalam mesh yang dibuat akun, tetapi mereka tidak dapat memodifikasi sumber daya yang dibuat akun lain.
- Konsumen dapat menghapus sumber daya apa pun dalam mesh yang dibuat akun.
- Pemilik dapat menghapus sumber daya apa pun di mesh yang dibuat akun apa pun.
- Sumber daya pemilik hanya dapat mereferensikan sumber daya lain di akun yang sama. Misalnya, node virtual hanya dapat referensi AWS Cloud Map atau AWS Certificate Manager sertifikat yang ada di akun yang sama dengan pemilik node virtual.
- Pemilik dan konsumen dapat menghubungkan proxy Utusan ke App Mesh sebagai simpul virtual yang dimiliki akun tersebut.
- Pemilik dapat membuat gateway virtual dan rute gateway virtual.
- Pemilik dan konsumen dapat membuat daftar tag dan dapat tag/untag sumber daya dalam mesh yang dibuat akun. Mereka tidak dapat mencantumkan tag dan tag/untag sumber daya di mesh yang tidak dibuat oleh akun.

Shared mesh menggunakan otorisasi berbasis kebijakan. Mesh dibagikan dengan satu set izin tetap. Izin ini dipilih untuk ditambahkan ke kebijakan sumber daya, dan kebijakan IAM opsional juga dapat dipilih berdasarkan pengguna/peran IAM. Perpotongan izin yang diizinkan dalam kebijakan ini, apalagi izin eksplisit yang ditolak, menentukan akses prinsipal ke mesh.

Saat berbagi mesh, AWS Resource Access Manager layanan akan membuat kebijakan terkelola bernama `AWSRAMDefaultPermissionAppMesh` dan mengaitkannya dengan App Mesh Anda yang menyediakan izin berikut.

- `appmesh:CreateVirtualNode`
- `appmesh:CreateVirtualRouter`
- `appmesh:CreateRoute`
- `appmesh:CreateVirtualService`
- `appmesh:UpdateVirtualNode`
- `appmesh:UpdateVirtualRouter`
- `appmesh:UpdateRoute`
- `appmesh:UpdateVirtualService`
- `appmesh:ListVirtualNodes`
- `appmesh:ListVirtualRouters`
- `appmesh:ListRoutes`
- `appmesh:ListVirtualServices`
- `appmesh:DescribeMesh`
- `appmesh:DescribeVirtualNode`
- `appmesh:DescribeVirtualRouter`
- `appmesh:DescribeRoute`
- `appmesh:DescribeVirtualService`
- `appmesh>DeleteVirtualNode`
- `appmesh>DeleteVirtualRouter`
- `appmesh>DeleteRoute`
- `appmesh>DeleteVirtualService`
- `appmesh:TagResource`
- `appmesh:UntagResource`

Prasyarat untuk berbagi jerat

Untuk berbagi jala, Anda harus memenuhi prasyarat berikut.

- Anda harus memiliki mesh di AWS akun Anda. Anda tidak dapat berbagi mesh yang telah dibagikan dengan Anda.
- Untuk berbagi mesh dengan organisasi Anda atau unit organisasi di AWS Organizations, Anda harus mengaktifkan berbagi dengan AWS Organizations. Untuk informasi selengkapnya, lihat [Mengaktifkan Berbagi dengan AWS Organizations](#) di Panduan AWS RAM Pengguna.
- Layanan Anda harus digunakan di VPC Amazon yang memiliki konektivitas bersama di seluruh akun yang menyertakan sumber daya mesh yang ingin Anda komunikasikan satu sama lain. Salah satu cara untuk berbagi konektivitas jaringan adalah dengan menyebarkan semua layanan yang ingin Anda gunakan di mesh Anda ke subnet bersama. Untuk informasi dan batasan selengkapnya, lihat [Berbagi Subnet](#).
- Layanan harus dapat ditemukan melalui DNS atau AWS Cloud Map. Untuk informasi selengkapnya tentang penemuan layanan, lihat [Node virtual](#).

Layanan terkait

Berbagi mesh terintegrasi dengan AWS Resource Access Manager (AWS RAM). AWS RAM adalah layanan yang memungkinkan Anda untuk berbagi AWS sumber daya Anda dengan AWS akun apa pun atau melalui AWS Organizations. Dengan AWS RAM, Anda berbagi sumber daya yang Anda miliki dengan membuat pembagian sumber daya. Pembagian sumber daya menentukan sumber daya yang akan dibagikan, dan konsumen yang akan dibagikan. Konsumen dapat berupa AWS akun individu, atau unit organisasi atau seluruh organisasi di dalamnya AWS Organizations.

Untuk informasi selengkapnya AWS RAM, lihat [Panduan AWS RAM Pengguna](#).

Berbagi jala

Berbagi mesh memungkinkan sumber daya mesh yang dibuat oleh akun yang berbeda untuk berkomunikasi satu sama lain dalam mesh yang sama. Anda hanya dapat berbagi mesh yang Anda miliki. Untuk berbagi mesh, Anda harus menambahkannya ke berbagi sumber daya. Berbagi sumber daya adalah AWS RAM sumber daya yang memungkinkan Anda berbagi sumber daya di seluruh AWS akun. Pembagian sumber daya menentukan sumber daya untuk dibagikan dan konsumen dengan siapa mereka dibagikan. Saat Anda berbagi mesh menggunakan konsol Amazon Linux, Anda menambahkannya ke pembagian sumber daya yang ada. Untuk menambahkan mesh ke pembagian sumber daya baru, buat pembagian sumber daya menggunakan [AWS RAM konsol](#).

Jika Anda bagian dari organisasi AWS Organizations dan berbagi dalam organisasi Anda diaktifkan, konsumen di organisasi Anda dapat secara otomatis diberikan akses ke mesh bersama. Jika tidak,

konsumen menerima undangan untuk bergabung dengan pembagian sumber daya dan diberikan akses ke mesh bersama setelah menerima undangan.

Anda dapat berbagi mesh yang Anda miliki menggunakan AWS RAM konsol atau AWS CLI.

Untuk berbagi mesh yang Anda miliki menggunakan AWS RAM konsol

Untuk petunjuknya, lihat [Membuat Pembagian Sumber Daya](#) di Panduan AWS RAM Pengguna. Saat Anda memilih jenis sumber daya, pilih Meshes, lalu pilih mesh yang ingin Anda bagikan. Jika tidak ada jerat yang terdaftar, buat mesh terlebih dahulu. Untuk informasi selengkapnya, lihat [Membuat mesh layanan](#).

Untuk berbagi mesh yang Anda miliki menggunakan AWS CLI

Gunakan perintah [create-resource-share](#). Untuk `--resource-arns` opsi, tentukan ARN dari mesh yang ingin Anda bagikan.

Membatalkan berbagi mesh bersama

Saat Anda melepaskan jala, App Mesh menonaktifkan akses lebih lanjut ke mesh oleh mantan konsumen mesh. Namun, App Mesh tidak menghapus sumber daya yang dibuat oleh konsumen. Setelah mesh tidak dibagi, hanya pemilik mesh yang dapat mengakses dan menghapus sumber daya. App Mesh mencegah akun yang memiliki sumber daya di mesh menerima informasi konfigurasi setelah mesh tidak dibagikan. App Mesh juga mencegah akun lain dengan sumber daya di mesh menerima informasi konfigurasi dari mesh yang tidak dibagikan. Hanya pemilik jala yang dapat membatalkan bagiannya.

Untuk membatalkan berbagi mesh bersama yang Anda miliki, Anda harus menghapusnya dari pembagian sumber daya. Anda dapat melakukan ini menggunakan AWS RAM konsol atau AWS CLI.

Untuk membatalkan berbagi mesh bersama yang Anda miliki menggunakan konsol AWS RAM

Untuk petunjuk, lihat [Memperbarui Pembagian Sumber Daya](#) di Panduan AWS RAM Pengguna.

Untuk membatalkan berbagi mesh bersama yang Anda miliki menggunakan AWS CLI

Gunakan perintah [disassociate-resource-share](#).

Mengidentifikasi mesh bersama

Pemilik dan konsumen dapat mengidentifikasi mesh bersama dan sumber daya mesh menggunakan konsol Amazon Linux dan AWS CLI

Untuk mengidentifikasi mesh bersama menggunakan konsol Amazon Linux

1. Buka konsol App Mesh di <https://console.aws.amazon.com/appmesh/>.
2. Dari navigasi kiri, pilih Meshes. ID akun pemilik mesh untuk setiap mesh tercantum di kolom pemilik Mesh.
3. Dari navigasi kiri, pilih Layanan virtual, router virtual, atau node Virtual. Anda melihat ID akun untuk pemilik Mesh dan pemilik Sumber Daya untuk setiap sumber daya.

Untuk mengidentifikasi mesh bersama menggunakan AWS CLI

Gunakan `aws appmesh list resource` perintah, seperti `aws appmesh list-meshes`. Perintah mengembalikan jerat yang Anda miliki dan jerat yang dibagikan dengan Anda. `meshOwnerProperti` menampilkan ID AWS akun `meshOwner` dan `resourceOwner` properti menunjukkan ID AWS akun pemilik sumber daya. Perintah apa pun yang dijalankan terhadap sumber daya mesh apa pun mengembalikan properti ini.

Tag yang ditentukan pengguna yang Anda lampirkan ke mesh bersama hanya tersedia untuk Anda Akun AWS. Mereka tidak tersedia untuk akun lain yang digunakan bersama mesh. `aws appmesh list-tags-for-resource` Perintah untuk mesh di akun lain ditolak aksesnya.

Tagihan dan pengukuran

Tidak ada biaya untuk berbagi jala.

Kuota contoh

Semua kuota untuk mesh juga berlaku untuk jerat bersama, terlepas dari siapa yang membuat sumber daya di mesh. Hanya pemilik mesh yang dapat meminta kenaikan kuota. Untuk informasi selengkapnya, lihat [Kuota layanan App Mesh](#). AWS Resource Access Manager Layanan ini juga memiliki kuota. Untuk informasi selengkapnya, lihat [Service Quotas](#).

AWS layanan terintegrasi dengan App Mesh

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk. AWS App Mesh Setelah 30 September 2026, Anda tidak akan lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

App Mesh bekerja dengan AWS layanan lain untuk memberikan solusi tambahan untuk tantangan bisnis Anda. Topik ini mengidentifikasi layanan yang menggunakan App Mesh untuk menambahkan fungsionalitas, atau layanan yang digunakan App Mesh untuk melakukan tugas.

Daftar Isi

- [Membuat sumber daya App Mesh dengan AWS CloudFormation](#)
- [App Mesh di AWS Outposts](#)

Membuat sumber daya App Mesh dengan AWS CloudFormation

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk. AWS App Mesh Setelah 30 September 2026, Anda tidak akan lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

App Mesh 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 mesh App Mesh, dan CloudFormation menangani penyediaan dan konfigurasi sumber daya tersebut untuk Anda.

Saat menggunakan CloudFormation, Anda dapat menggunakan kembali template untuk menyiapkan resource App Mesh secara konsisten dan berulang kali. Cukup jelaskan sumber daya Anda sekali, lalu sediakan sumber daya yang sama berulang-ulang di beberapa AWS akun dan Wilayah.

App Mesh dan CloudFormation template

Untuk menyediakan dan mengonfigurasi sumber daya untuk App Mesh dan layanan terkait, Anda harus memahami [CloudFormation templat](#). Templat adalah file teks dengan format JSON atau YAML. Template ini menjelaskan sumber daya yang ingin Anda sediakan di CloudFormation tumpukan Anda. Jika Anda tidak terbiasa dengan JSON atau YAMG, Anda dapat menggunakan CloudFormation Designer untuk membantu Anda memulai dengan template. CloudFormation Untuk informasi lebih lanjut, lihat [Apa itu CloudFormation Desainer?](#) dalam AWS CloudFormation User Guide.

App Mesh mendukung pembuatan jerat, rute, node virtual, router virtual, dan layanan virtual di CloudFormation Untuk informasi selengkapnya, termasuk contoh template JSON dan YAMG untuk resource App Mesh Anda, lihat [referensi tipe resource App Mesh](#) di AWS CloudFormation Panduan Pengguna.

Pelajari lebih lanjut tentang CloudFormation

Untuk mempelajari selengkapnya CloudFormation, lihat sumber daya berikut:

- [AWS CloudFormation](#)
- [AWS CloudFormation Panduan Pengguna](#)
- [AWS CloudFormation Panduan Pengguna Antarmuka Baris Perintah](#)

App Mesh di AWS Outposts

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk. AWS App Mesh Setelah 30 September 2026, Anda tidak akan lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

AWS Outposts memungkinkan AWS layanan asli, infrastruktur, dan model operasi di fasilitas lokal. Di lingkungan AWS Outposts, Anda dapat menggunakan alat AWS APIs, dan infrastruktur yang sama yang Anda gunakan di AWS Cloud. App Mesh di AWS Outposts sangat ideal untuk beban kerja latensi rendah yang perlu dijalankan di dekat data dan aplikasi lokal. Untuk informasi selengkapnya tentang AWS Outposts, lihat Panduan Pengguna [AWS Outposts](#).

Prasyarat

Berikut ini adalah prasyarat untuk menggunakan App Mesh di Outposts: AWS

- Anda harus menginstal dan mengonfigurasi Outpost di pusat data On-Premise Anda.
- Anda harus memiliki koneksi jaringan yang dapat diandalkan antara Outpost Anda dan Wilayah AWS .
- AWS Wilayah untuk Pos Terdepan harus mendukung AWS App Mesh. Untuk daftar Wilayah yang didukung, lihat [AWS App Mesh Titik Akhir dan Kuota](#) di. Referensi Umum AWS

Batasan

Berikut ini adalah batasan penggunaan App Mesh di AWS Outposts:

- AWS Identity and Access Management, Application Load Balancer, Network Load Balancer, Classic Load Balancer, dan Amazon Route 53 berjalan di AWS Wilayah, bukan di Outposts. Hal ini akan meningkatkan latensi antara layanan ini dan kontainer.

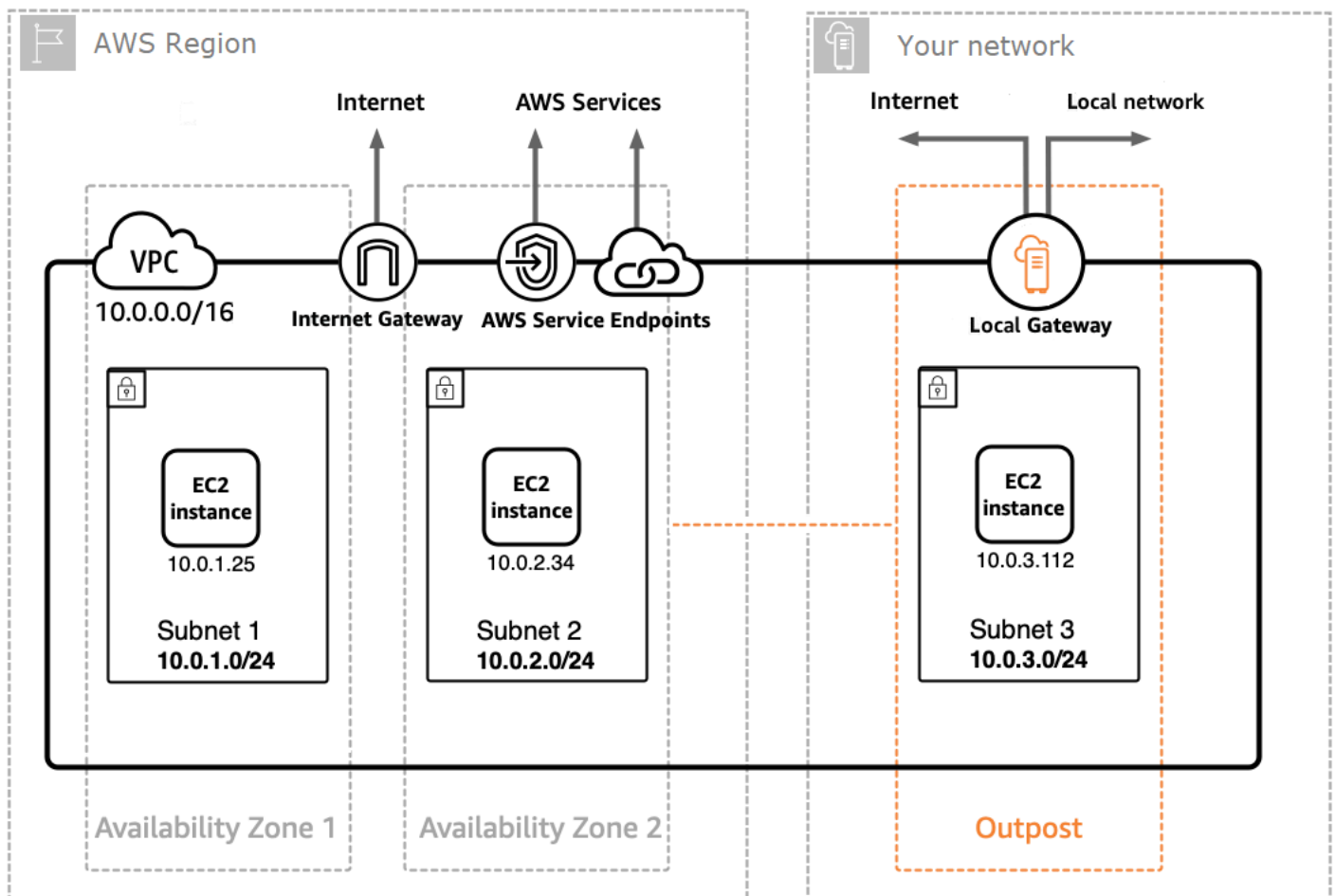
Pertimbangan konektivitas jaringan

Berikut ini adalah pertimbangan konektivitas jaringan untuk Amazon EKS AWS Outposts:

- Jika konektivitas jaringan antara Outpost Anda dan AWS Wilayahnya hilang, proxy App Mesh Envoy akan terus berjalan. Namun Anda tidak akan dapat memodifikasi mesh layanan Anda sampai konektivitas dipulihkan.
- Kami menyarankan Anda menyediakan konektivitas yang andal, sangat tersedia, dan latensi rendah antara Outpost Anda dan Wilayahnya. AWS

Membuat proxy App Mesh Envoy di Outpost

Pos Luar adalah perpanjangan dari AWS Wilayah, dan Anda dapat memperluas VPC Amazon di akun untuk menjangkau beberapa Availability Zone dan lokasi Outpost terkait. Saat Anda mengonfigurasi Outpost, Anda mengaitkan subnet dengannya untuk memperluas lingkungan VPC Regional Anda ke fasilitas on premise. Instance di Outpost muncul sebagai bagian dari VPC Regional Anda, mirip dengan Availability Zone dengan subnet terkait.



Untuk membuat proxy App Mesh Envoy di Outpost, tambahkan image container App Mesh Envoy ke tugas Amazon ECS atau pod Amazon EKS yang berjalan di Outpost. Untuk informasi selengkapnya, lihat [Amazon Elastic Container Service on AWS Outposts](#) di Amazon Elastic Container Service Developer Guide dan Amazon Elastic Kubernetes Service AWS on Outposts di Panduan [Pengguna Amazon EKS](#).

Praktik terbaik App Mesh

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk AWS App Mesh. Setelah 30 September 2026, Anda tidak akan lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Untuk mencapai tujuan nol permintaan yang gagal selama penerapan yang direncanakan dan selama kehilangan beberapa host yang tidak direncanakan, praktik terbaik dalam topik ini menerapkan strategi berikut:

- Tingkatkan kemungkinan permintaan akan berhasil dari perspektif aplikasi dengan menggunakan strategi coba ulang default yang aman. Untuk informasi selengkapnya, lihat [Instrumen semua rute dengan percobaan ulang](#).
- Tingkatkan kemungkinan permintaan yang dicoba ulang berhasil dengan memaksimalkan kemungkinan permintaan yang dicoba ulang dikirim ke tujuan yang sebenarnya. Lihat informasi selengkapnya di [Sesuaikan kecepatan penyebaran](#), [Skalakan sebelum skala di](#), dan [Melaksanakan pemeriksaan kesehatan kontainer](#).

Untuk mengurangi atau menghilangkan kegagalan secara signifikan, kami sarankan Anda menerapkan rekomendasi dalam semua praktik berikut.

Instrumen semua rute dengan percobaan ulang

Konfigurasi semua layanan virtual untuk menggunakan router virtual dan tetapkan kebijakan coba ulang default untuk semua rute. Ini akan mengurangi permintaan yang gagal dengan memilih ulang host dan mengirim permintaan baru. Untuk kebijakan coba lagi, kami merekomendasikan nilai minimal dua untuk `maxRetries`, dan menentukan opsi berikut untuk setiap jenis peristiwa coba lagi di setiap jenis rute yang mendukung jenis peristiwa coba lagi:

- TCP — `connection-error`
- HTTP dan HTTP/2 — `stream-error gateway-error`

- gRPC — dan cancelled unavailable

Peristiwa coba lagi lainnya perlu dipertimbangkan karena case-by-case mungkin tidak aman, seperti jika permintaan tidak idempoten. Anda perlu mempertimbangkan dan menguji nilai untuk `maxRetries` dan `perRetryTimeout` yang membuat trade off yang tepat antara latensi maksimum permintaan ($\text{maxRetries} \times \text{perRetryTimeout}$) versus tingkat keberhasilan yang meningkat dari lebih banyak percobaan ulang. Selain itu, ketika Envoy mencoba untuk terhubung ke titik akhir yang tidak lagi ada, Anda harus mengharapkan permintaan itu untuk dikonsumsi penuh. `perRetryTimeout` Untuk mengonfigurasi kebijakan coba lagi, lihat [Membuat rute](#) lalu pilih protokol yang ingin Anda rutekan.

Note

Jika Anda menerapkan rute pada atau setelah 29 Juli 2020 dan tidak menentukan kebijakan coba lagi, App Mesh mungkin telah secara otomatis membuat kebijakan coba ulang default yang serupa dengan kebijakan sebelumnya untuk setiap rute yang Anda buat pada atau setelah 29 Juli 2020. Untuk informasi selengkapnya, lihat [Kebijakan coba lagi rute default](#).

Sesuaikan kecepatan penyebaran

Saat menggunakan penerapan bergulir, kurangi kecepatan penyebaran secara keseluruhan. Secara default, Amazon ECS mengonfigurasi strategi penyebaran minimal 100 persen tugas sehat dan 200 persen total tugas. Pada penerapan, ini menghasilkan dua titik penyimpangan tinggi:

- Ukuran armada tugas baru 100 persen mungkin terlihat oleh Utusan sebelum siap untuk menyelesaikan permintaan (lihat [Melaksanakan pemeriksaan kesehatan kontainer](#) untuk mitigasi).
- Ukuran armada 100 persen dari tugas-tugas lama mungkin terlihat oleh Utusan saat tugas sedang dihentikan.

Ketika dikonfigurasi dengan batasan penerapan ini, orkestra kontainer dapat memasuki status di mana mereka secara bersamaan menyembunyikan semua tujuan lama dan membuat semua tujuan baru terlihat. Karena jalur data Utusan Anda pada akhirnya konsisten, ini dapat menghasilkan periode di mana kumpulan tujuan yang terlihat di jalur data Anda telah menyimpang dari sudut pandang orkestrator. Untuk mengurangi ini, kami sarankan mempertahankan minimal 100 persen tugas sehat, tetapi menurunkan total tugas menjadi 125 persen. Ini akan mengurangi divergensi

dan meningkatkan keandalan percobaan ulang. Kami merekomendasikan pengaturan berikut untuk runtime kontainer yang berbeda:

Amazon ECS

Jika layanan Anda memiliki hitungan dua atau tiga yang diinginkan, atur `maximumPercent` ke 150 persen. Jika tidak, atur `maximumPercent` ke 125 persen.

Kubernetes

Konfigurasi penerapan Anda `update strategy`, setel `maxUnavailable` ke 0 persen dan `maxSurge` hingga 25 persen. [Untuk informasi selengkapnya tentang penerapan, lihat dokumentasi Penerapan Kubernetes.](#)

Skalakan sebelum skala di

Scale out dan scale in keduanya dapat menghasilkan beberapa kemungkinan permintaan gagal dalam percobaan ulang. Meskipun ada rekomendasi tugas yang mengurangi skala, satu-satunya rekomendasi untuk skala adalah meminimalkan persentase tugas yang diskalakan pada satu waktu. Sebaiknya gunakan strategi penerapan yang mengukur tugas Amazon ECS baru atau penerapan Kubernetes sebelum melakukan penskalaan pada tugas atau penerapan lama. Strategi penskalaan ini menjaga persentase penskalaan tugas atau penerapan Anda lebih rendah, sambil mempertahankan kecepatan yang sama. Praktik ini berlaku untuk tugas Amazon ECS dan penerapan Kubernetes.

Melaksanakan pemeriksaan kesehatan kontainer

Dalam skenario peningkatan skala, kontainer dalam tugas Amazon ECS mungkin rusak dan mungkin pada awalnya tidak responsif. Kami merekomendasikan saran berikut untuk runtime kontainer yang berbeda:

Amazon ECS

Untuk mengurangi hal ini, sebaiknya gunakan pemeriksaan kesehatan kontainer dan pemesanan ketergantungan kontainer untuk memastikan bahwa Utusan berjalan dan sehat sebelum kontainer apa pun yang memerlukan konektivitas jaringan keluar dimulai. Untuk mengonfigurasi wadah aplikasi dan wadah Envoy dengan benar dalam definisi tugas, lihat Ketergantungan [kontainer](#).

Kubernetes

[Tidak ada, karena probe keaktifan dan kesiapan Kubernetes tidak dipertimbangkan dalam registrasi dan de-registrasi instance di AWS Cloud Map pengontrol App Mesh untuk Kubernetes.](#) Untuk informasi selengkapnya, lihat GitHub edisi [#132](#).

Optimalkan resolusi DNS

Jika Anda menggunakan DNS untuk penemuan layanan, penting untuk memilih protokol IP yang sesuai untuk mengoptimalkan resolusi DNS saat mengonfigurasi jerat Anda. App Mesh mendukung keduanya IPv4 dan IPv6, dan pilihan Anda dapat memengaruhi kinerja dan kompatibilitas layanan Anda. Jika infrastruktur Anda tidak mendukung IPv6, kami sarankan Anda menentukan setelan IP yang selaras dengan infrastruktur Anda daripada mengandalkan perilaku default IPv6_PREFERRED. IPv6_PREFERRED Perilaku default dapat menurunkan kinerja layanan.

- **IPv6_PREFERRED** — Ini adalah pengaturan default. Utusan melakukan pencarian DNS untuk IPv6 alamat terlebih dahulu dan kembali ke IPv4 jika tidak ada IPv6 alamat yang ditemukan. Ini bermanfaat jika infrastruktur Anda terutama mendukung IPv6 tetapi membutuhkan IPv4 kompatibilitas.
- **IPv4_PREFERRED** — Utusan pertama mencari IPv4 alamat dan kembali ke IPv6 jika tidak ada IPv4 alamat yang tersedia. Gunakan pengaturan ini jika infrastruktur Anda terutama mendukung IPv4 tetapi memiliki beberapa IPv6 kompatibilitas.
- **IPv6_ONLY** — Pilih opsi ini jika layanan Anda secara eksklusif mendukung IPv6 lalu lintas. Utusan hanya melakukan pencarian DNS untuk IPv6 alamat, memastikan semua lalu lintas dialihkan. IPv6
- **IPv4_ONLY** — Pilih pengaturan ini jika layanan Anda secara eksklusif mendukung IPv4 lalu lintas. Utusan hanya melakukan pencarian DNS untuk IPv4 alamat, memastikan semua lalu lintas dialihkan. IPv4

Anda dapat mengatur preferensi versi IP pada level mesh dan tingkat node virtual, dengan pengaturan node virtual mengesampingkan yang ada di level mesh.

Untuk informasi selengkapnya, lihat [Service Meshes](#) dan [Virtual Nodes](#).

Keamanan di AWS App Mesh

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk AWS App Mesh. Setelah 30 September 2026, Anda tidak akan lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

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](#) menggambarkan hal ini sebagai keamanan dari cloud dan keamanan di cloud:

- Keamanan cloud — AWS bertanggung jawab untuk melindungi infrastruktur yang menjalankan AWS layanan di AWS Cloud. AWS juga memberi Anda layanan yang dapat Anda gunakan dengan aman. Auditor pihak ketiga menguji dan memverifikasi efektivitas keamanan kami sebagai bagian dari [program kepatuhan AWS](#). Untuk mempelajari program kepatuhan yang berlaku di AWS App Mesh, lihat [Cakupan Layanan Menurut Program Kepatuhan AWS](#). App Mesh bertanggung jawab untuk mengirimkan konfigurasi secara aman ke proxy lokal, termasuk rahasia seperti kunci pribadi sertifikat TLS.
- Keamanan di cloud — Tanggung jawab Anda ditentukan oleh AWS layanan yang Anda gunakan. Anda juga bertanggung jawab atas faktor-faktor lain termasuk:
 - Sensitivitas data Anda, persyaratan perusahaan Anda, dan hukum dan peraturan yang berlaku.
 - Konfigurasi keamanan pesawat data App Mesh, termasuk konfigurasi grup keamanan yang memungkinkan lalu lintas lewat antar layanan dalam VPC Anda.
 - Konfigurasi sumber daya komputasi Anda yang terkait dengan App Mesh.
 - Kebijakan IAM yang terkait dengan sumber daya komputasi Anda dan konfigurasi apa yang diizinkan untuk diambil dari bidang kontrol App Mesh.

Dokumentasi ini membantu Anda memahami cara menerapkan model tanggung jawab bersama saat menggunakan App Mesh. Topik berikut menunjukkan cara mengonfigurasi App Mesh untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga mempelajari cara menggunakan AWS layanan lain yang membantu Anda memantau dan mengamankan sumber daya App Mesh Anda.

Prinsip keamanan App Mesh

Pelanggan harus dapat menyesuaikan keamanan sejauh yang mereka butuhkan. Platform seharusnya tidak menghalangi mereka agar tidak lebih aman. Fitur platform aman secara default.

Topik

- [Keamanan Lapisan Pengangkutan \(TLS\)](#)
- [Autentikasi TLS bersama](#)
- [Bagaimana AWS App Mesh bekerja dengan IAM](#)
- [Logging panggilan AWS App Mesh API menggunakan AWS CloudTrail](#)
- [Perlindungan data di AWS App Mesh](#)
- [Validasi kepatuhan untuk AWS App Mesh](#)
- [Keamanan infrastruktur di AWS App Mesh](#)
- [Ketahanan di AWS App Mesh](#)
- [Konfigurasi dan analisis kerentanan di AWS App Mesh](#)

Keamanan Lapisan Pengangkutan (TLS)

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk AWS App Mesh. Setelah 30 September 2026, Anda tidak akan lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Di App Mesh, Transport Layer Security (TLS) mengenkripsi komunikasi antara proxy Envoy yang digunakan pada sumber daya komputasi yang direpresentasikan dalam App Mesh oleh titik akhir mesh, seperti [Node virtual](#) dan [Gerbang virtual](#). Proxy menegosiasikan dan mengakhiri TLS. Ketika

proxy digunakan dengan aplikasi, kode aplikasi Anda tidak bertanggung jawab untuk menegosiasikan sesi TLS. Proxy menegosiasikan TLS atas nama aplikasi Anda.

App Mesh memungkinkan Anda memberikan sertifikat TLS ke proxy dengan cara berikut:

- Sertifikat pribadi dari AWS Certificate Manager (ACM) yang dikeluarkan oleh AWS Private Certificate Authority (AWS Private CA).
- Sertifikat yang disimpan pada sistem file lokal dari node virtual yang dikeluarkan oleh otoritas sertifikat (CA) Anda sendiri
- Sertifikat yang disediakan oleh titik akhir Secrets Discovery Service (SDS) di atas Unix Domain Socket lokal.

[Otorisasi Proxy Utusan](#) harus diaktifkan untuk proxy Utusan yang diterapkan yang diwakili oleh titik akhir mesh. Kami menyarankan bahwa ketika Anda mengaktifkan otorisasi proxy, Anda membatasi akses hanya ke titik akhir mesh yang Anda aktifkan enkripsi.

Persyaratan sertifikat

Salah satu Nama Alternatif Subjek (SANs) pada sertifikat harus sesuai dengan kriteria tertentu, tergantung pada bagaimana layanan aktual yang diwakili oleh titik akhir mesh ditemukan.

- DNS — Salah satu sertifikat SANs harus sesuai dengan nilai yang diberikan dalam pengaturan penemuan layanan DNS. Untuk aplikasi dengan nama penemuan layanan `mesh-endpoint.apps.local`, Anda dapat membuat sertifikat yang cocok dengan nama itu, atau sertifikat dengan kartu liar* `.apps.local`.
- AWS Cloud Map— Salah satu sertifikat SANs harus sesuai dengan nilai yang diberikan dalam pengaturan penemuan AWS Cloud Map layanan menggunakan format `service-name.namespace-name`. Untuk aplikasi dengan pengaturan penemuan AWS Cloud Map layanan `ServiceName` dan `mesh-endpoint apps.local` `NamespaceName`, Anda dapat membuat sertifikat yang cocok dengan `mesh-endpoint.apps.local` nama, atau sertifikat dengan kartu liar * `.apps.local`.

Untuk kedua mekanisme penemuan, jika tidak ada sertifikat yang SANs cocok dengan pengaturan penemuan layanan DNS, koneksi antara Utusan gagal dengan pesan galat berikut, seperti yang terlihat dari Utusan klien.

```
TLS error: 268435581:SSL routines:OPENSSL_internal:CERTIFICATE_VERIFY_FAILED
```

Sertifikat otentikasi TLS

App Mesh mendukung beberapa sumber untuk sertifikat saat menggunakan otentikasi TLS.

AWS Private CA

Sertifikat harus disimpan di ACM di Wilayah yang sama dan AWS akun sebagai titik akhir mesh yang akan menggunakan sertifikat. Sertifikat CA tidak harus berada di AWS akun yang sama, tetapi masih harus berada di Wilayah yang sama dengan titik akhir mesh. Jika Anda tidak memiliki AWS Private CA, maka Anda harus [membuatnya](#) sebelum Anda dapat meminta sertifikat darinya. Untuk informasi selengkapnya tentang meminta sertifikat dari ACM yang sudah ada AWS Private CA, lihat [Meminta Sertifikat Pribadi](#). Sertifikat tidak dapat berupa sertifikat publik.

Privat CAs yang Anda gunakan untuk kebijakan klien TLS harus pengguna CAs root.

Untuk mengonfigurasi node virtual dengan sertifikat dan CAs dari AWS Private CA, prinsipal (seperti pengguna atau peran) yang Anda gunakan untuk memanggil App Mesh harus memiliki izin IAM berikut:

- Untuk sertifikat apa pun yang Anda tambahkan ke konfigurasi TLS pendengar, kepala sekolah harus memiliki izin. `acm:DescribeCertificate`
- Untuk setiap yang CAs dikonfigurasi pada kebijakan klien TLS, kepala sekolah harus memiliki `acm-pca:DescribeCertificateAuthority` izin.

Important

Berbagi CAs dengan akun lain dapat memberikan akun-akun tersebut hak istimewa yang tidak diinginkan kepada CA. Sebaiknya gunakan kebijakan berbasis sumber daya untuk membatasi akses ke akun yang adil `acm-pca:DescribeCertificateAuthority` dan `acm-pca:GetCertificateAuthorityCertificate` untuk akun yang tidak perlu mengeluarkan sertifikat dari CA.

Anda dapat menambahkan izin ini ke kebijakan IAM yang sudah ada yang dilampirkan pada prinsipal atau membuat prinsipal dan kebijakan baru dan melampirkan kebijakan tersebut ke prinsipal. Untuk informasi selengkapnya, lihat [Mengedit Kebijakan IAM, Membuat Kebijakan IAM](#), dan [Menambahkan Izin Identitas IAM](#).

Note

Anda membayar biaya bulanan untuk pengoperasian masing-masing AWS Private CA sampai Anda menghapusnya. Anda juga membayar sertifikat pribadi yang Anda terbitkan setiap bulan dan sertifikat pribadi yang Anda ekspor. Untuk informasi selengkapnya, silakan lihat [Harga AWS Certificate Manager](#).

Saat Anda mengaktifkan [otorisasi proxy](#) untuk Proxy Utusan yang diwakili oleh titik akhir mesh, peran IAM yang Anda gunakan harus diberi izin IAM berikut:

- Untuk sertifikat apa pun yang dikonfigurasi pada pendengar node virtual, peran harus memiliki `acm:ExportCertificate` izin.
- Untuk setiap yang CAs dikonfigurasi pada kebijakan klien TLS, peran harus memiliki `acm-pca:GetCertificateAuthorityCertificate` izin.

Sistem file

Anda dapat mendistribusikan sertifikat ke Utusan menggunakan sistem file. Anda dapat melakukan ini dengan membuat rantai sertifikat dan kunci pribadi yang sesuai tersedia di jalur file. Dengan begitu, sumber daya ini dapat dijangkau dari proxy sespan Utusan.

Layanan Penemuan Rahasia Utusan (SDS)

Utusan mengambil rahasia seperti sertifikat TLS dari titik akhir tertentu melalui protokol Secrets Discovery. Untuk informasi selengkapnya tentang protokol ini, lihat dokumentasi [SDS Envoy](#).

App Mesh mengonfigurasi proxy Envoy untuk menggunakan Unix Domain Socket yang lokal ke proxy untuk berfungsi sebagai titik akhir Secret Discovery Service (SDS) saat SDS berfungsi sebagai sumber sertifikat dan rantai sertifikat Anda. Anda dapat mengonfigurasi jalur ke titik akhir ini dengan menggunakan variabel `APPMESH_SDS_SOCKET_PATH` lingkungan.

Important

Layanan Penemuan Rahasia Lokal menggunakan Unix Domain Socket didukung pada proxy App Mesh Envoy versi 1.15.1.0 dan yang lebih baru.
App Mesh mendukung protokol V2 SDS menggunakan gRPC.

Mengintegrasikan dengan SPIFFE Runtime Environment (SPIRE)

Anda dapat menggunakan implementasi sidecar apa pun dari SDS API, termasuk toolchain yang ada seperti [SPIFFE Runtime Environment \(SPIRE\)](#). SPIRE dirancang untuk memungkinkan penyebaran otentikasi TLS timbal balik antara beberapa beban kerja dalam sistem terdistribusi. Ini membuktikan identitas beban kerja saat runtime. SPIRE juga memberikan kunci dan sertifikat khusus beban kerja, berumur pendek, dan otomatis berputar langsung ke beban kerja.

Anda harus mengonfigurasi Agen SPIRE sebagai penyedia SDS untuk Utusan. Izinkan untuk secara langsung memasok Utusan dengan materi utama yang dibutuhkan untuk memberikan otentikasi TLS timbal balik. Jalankan SPIRE Agents di sidecars di sebelah proxy Utusan. Agen menangani pembuatan kembali kunci dan sertifikat berumur pendek sesuai kebutuhan. Agen membuktikan Utusan dan menentukan identitas layanan dan sertifikat CA mana yang harus disediakan untuk Utusan ketika Utusan terhubung ke server SDS yang diekspos oleh Agen SPIRE.

Selama proses ini, identitas layanan dan sertifikat CA diputar, dan pembaruan dialirkan kembali ke Envoy. Utusan segera menerapkannya ke koneksi baru tanpa gangguan atau downtime dan tanpa kunci pribadi yang pernah menyentuh sistem file.

Bagaimana App Mesh mengonfigurasi Utusan untuk menegosiasikan TLS

App Mesh menggunakan konfigurasi titik akhir mesh dari klien dan server saat menentukan cara mengonfigurasi komunikasi antara Utusan dalam mesh.

Dengan kebijakan klien

Ketika kebijakan klien menerapkan penggunaan TLS, dan salah satu port dalam kebijakan klien cocok dengan port kebijakan server, kebijakan klien digunakan untuk mengonfigurasi konteks validasi TLS klien. Misalnya, jika kebijakan klien gateway virtual cocok dengan kebijakan server node virtual, negosiasi TLS akan dicoba antara proxy menggunakan pengaturan yang ditentukan dalam kebijakan klien gateway virtual. Jika kebijakan klien tidak cocok dengan port kebijakan server, TLS antara proxy mungkin atau mungkin tidak dinegosiasikan, tergantung pada pengaturan TLS kebijakan server.

Tanpa kebijakan klien

Jika klien belum mengonfigurasi kebijakan klien, atau kebijakan klien tidak cocok dengan port server, App Mesh akan menggunakan server untuk menentukan apakah akan menegosiasikan TLS dari klien atau tidak, dan bagaimana caranya. Misalnya, jika gateway virtual belum

menentukan kebijakan klien, dan node virtual belum mengonfigurasi penghentian TLS, TLS tidak akan dinegosiasikan antara proxy. Jika klien belum menentukan kebijakan klien yang cocok, dan server telah dikonfigurasi dengan mode TLS STRICT atau PERMISSIVE, proxy akan dikonfigurasi untuk menegosiasikan TLS. Bergantung pada bagaimana sertifikat telah disediakan untuk penghentian TLS, perilaku tambahan berikut berlaku.

- Sertifikat TLS yang dikelola ACM — Ketika server telah mengonfigurasi penghentian TLS menggunakan sertifikat yang dikelola ACM, App Mesh secara otomatis mengonfigurasi klien untuk menegosiasikan TLS dan memvalidasi sertifikat terhadap CA pengguna root yang dirantai sertifikat tersebut.
- Sertifikat TLS berbasis file — Ketika server telah mengonfigurasi penghentian TLS menggunakan sertifikat dari sistem file lokal proxy, App Mesh secara otomatis mengonfigurasi klien untuk menegosiasikan TLS, tetapi sertifikat server tidak divalidasi.

Nama alternatif subjek

Anda dapat secara opsional menentukan daftar Nama Alternatif Subjek (SANs) untuk dipercaya. SANs harus dalam format FQDN atau URI. Jika SANs disediakan, Utusan memverifikasi bahwa Nama Alternatif Subjek dari sertifikat yang disajikan cocok dengan salah satu nama dalam daftar ini.

Jika Anda tidak menentukan SANs titik akhir mesh terminating, proxy Envoy untuk node tersebut tidak memverifikasi SAN pada sertifikat klien rekan. Jika Anda tidak menentukan SANs titik akhir mesh asal, SAN pada sertifikat yang disediakan oleh titik akhir penghentian harus cocok dengan konfigurasi penemuan layanan titik akhir mesh.

Untuk informasi selengkapnya, lihat App Mesh [TLS: Persyaratan sertifikat](#).

Important

Anda hanya dapat menggunakan wildcard SANs jika kebijakan klien untuk TLS disetel ke `not_enforced`. Jika kebijakan klien untuk node virtual klien atau gateway virtual dikonfigurasi untuk menerapkan TLS, maka kebijakan tersebut tidak dapat menerima wildcard SAN.

Verifikasi enkripsi

Setelah mengaktifkan TLS, Anda dapat menanyakan proxy Utusan untuk mengonfirmasi bahwa komunikasi dienkripsi. Proxy utusan memancarkan statistik tentang sumber daya yang dapat

membantu Anda memahami apakah komunikasi TLS Anda berfungsi dengan baik. Misalnya, proxy Envoy mencatat statistik tentang jumlah jabat tangan TLS yang berhasil dinegosiasikan untuk titik akhir mesh tertentu. Tentukan berapa banyak jabat tangan TLS yang berhasil ada untuk titik akhir mesh bernama *my-mesh-endpoint* dengan perintah berikut.

```
curl -s 'http://my-mesh-endpoint.apps.local:9901/stats' | grep ssl.handshake
```

Dalam contoh berikut mengembalikan output, ada tiga jabat tangan untuk titik akhir mesh, sehingga komunikasi dienkripsi.

```
listener.0.0.0.0_15000.ssl.handshake: 3
```

Proxy utusan juga memancarkan statistik ketika negosiasi TLS gagal. Tentukan apakah ada kesalahan TLS untuk titik akhir mesh.

```
curl -s 'http://my-mesh-endpoint.apps.local:9901/stats' | grep -e "ssl.*\ (fail\|error\n)"
```

Dalam contoh output yang dikembalikan, tidak ada kesalahan untuk beberapa statistik, sehingga negosiasi TLS berhasil.

```
listener.0.0.0.0_15000.ssl.connection_error: 0
listener.0.0.0.0_15000.ssl.fail_verify_cert_hash: 0
listener.0.0.0.0_15000.ssl.fail_verify_error: 0
listener.0.0.0.0_15000.ssl.fail_verify_no_cert: 0
listener.0.0.0.0_15000.ssl.fail_verify_san: 0
```

Untuk informasi selengkapnya tentang statistik Envoy TLS, lihat Statistik Pendengar [Utusan](#).

Perpanjangan sertifikat

AWS Private CA

Saat Anda memperbarui sertifikat dengan ACM, sertifikat yang diperbarui akan didistribusikan secara otomatis ke proxy Anda yang terhubung dalam waktu 35 menit setelah perpanjangan selesai. Sebaiknya gunakan perpanjangan terkelola untuk memperbarui sertifikat secara otomatis mendekati akhir masa berlakunya. Untuk informasi selengkapnya, lihat [Pembaruan Terkelola untuk Sertifikat yang Diterbitkan Amazon ACM di Panduan Pengguna](#). AWS Certificate Manager

Sertifikat Anda sendiri

Saat menggunakan sertifikat dari sistem file lokal, Utusan tidak akan memuat ulang sertifikat secara otomatis saat berubah. Anda dapat memulai ulang atau menerapkan kembali proses Utusan untuk memuat sertifikat baru. Anda juga dapat menempatkan sertifikat yang lebih baru di jalur file yang berbeda dan memperbarui node virtual atau konfigurasi gateway dengan jalur file tersebut.

Konfigurasi beban kerja Amazon ECS untuk menggunakan otentikasi TLS AWS App Mesh

Anda dapat mengonfigurasi mesh Anda untuk menggunakan otentikasi TLS. Pastikan sertifikat tersedia untuk sidecar proxy Envoy yang Anda tambahkan ke beban kerja Anda. Anda dapat melampirkan volume EBS atau EFS ke sespan Envoy Anda, atau Anda dapat menyimpan dan mengambil sertifikat dari Secrets Manager. AWS

- Jika Anda menggunakan distribusi sertifikat berbasis file, lampirkan volume EBS atau EFS ke sespan Envoy Anda. Pastikan jalur ke sertifikat dan kunci pribadi cocok dengan yang dikonfigurasi AWS App Mesh.
- Jika Anda menggunakan distribusi berbasis SDS, tambahkan sespan yang mengimplementasikan SDS API Envoy dengan akses ke sertifikat.

Note

SPIRE tidak didukung di Amazon ECS.

Konfigurasi beban kerja Kubernetes untuk menggunakan otentikasi TLS dengan AWS App Mesh

Anda dapat mengonfigurasi AWS App Mesh Controller untuk Kubernetes untuk mengaktifkan otentikasi TLS untuk backend dan pendengar layanan virtual node dan gateway virtual. Pastikan sertifikat tersedia untuk sidecar proxy Envoy yang Anda tambahkan ke beban kerja Anda. Anda dapat melihat contoh untuk setiap jenis distribusi di bagian [panduan](#) Mutual TLS Authentication.

- Jika Anda menggunakan distribusi sertifikat berbasis file, lampirkan volume EBS atau EFS ke sespan Envoy Anda. Pastikan jalur ke sertifikat dan kunci pribadi cocok dengan yang dikonfigurasi di pengontrol. Atau, Anda dapat menggunakan Kubernetes Secret yang dipasang pada sistem file.
- Jika Anda menggunakan distribusi berbasis SDS, Anda harus menyiapkan penyedia SDS lokal node yang mengimplementasikan API SDS Envoy. Utusan akan mencapainya melalui UDS. Untuk

mengaktifkan dukungan mTL berbasis SDS di AppMesh pengontrol EKS, atur `enable-sds` flag ke `true` dan berikan jalur UDS penyedia SDS lokal ke pengontrol melalui bendera. `sds-uds-path` Jika Anda menggunakan helm, Anda mengatur ini sebagai bagian dari instalasi controller Anda:

```
--set sds.enabled=true
```

Note

Anda tidak akan dapat menggunakan SPIRE untuk mendistribusikan sertifikat Anda jika Anda menggunakan Amazon Elastic Kubernetes Service (Amazon EKS) dalam mode Fargate.

Autentikasi TLS bersama

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk AWS App Mesh. Setelah 30 September 2026, Anda tidak akan lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Otentikasi Mutual TLS (Transport Layer Security) adalah komponen opsional TLS yang menawarkan otentikasi peer dua arah. Mutual TLS otentikasi menambahkan lapisan keamanan melalui TLS dan memungkinkan layanan Anda untuk memverifikasi klien yang membuat koneksi.

Klien dalam hubungan client-server juga menyediakan sertifikat X.509 selama proses negosiasi sesi. Server menggunakan sertifikat ini untuk mengidentifikasi dan mengotentikasi klien. Proses ini membantu memverifikasi apakah sertifikat dikeluarkan oleh otoritas sertifikat tepercaya (CA) dan apakah sertifikat tersebut adalah sertifikat yang valid. Ini juga menggunakan Nama Alternatif Subjek (SAN) pada sertifikat untuk mengidentifikasi klien.

Anda dapat mengaktifkan otentikasi TLS timbal balik untuk semua protokol yang didukung oleh AWS App Mesh Mereka adalah TCP, HTTP/1.1, HTTP/2, gRPC.

Note

Menggunakan App Mesh, Anda dapat mengonfigurasi otentikasi TLS timbal balik untuk komunikasi antara proxy Utusan dari layanan Anda. Namun, komunikasi antara aplikasi Anda dan proxy Utusan tidak terenkripsi.

Sertifikat otentikasi TLS timbal balik

AWS App Mesh mendukung dua sumber sertifikat yang mungkin untuk otentikasi TLS timbal balik. Sertifikat klien dalam Kebijakan Klien TLS dan validasi server dalam konfigurasi TLS pendengar dapat bersumber dari:

- Sistem File— Sertifikat dari sistem file lokal proxy Utusan yang sedang dijalankan. Untuk mendistribusikan sertifikat ke Envoy, Anda harus menyediakan jalur file untuk rantai sertifikat dan kunci pribadi ke App Mesh API.
- Envoy's Secret Discovery Service (SDS) — Bring-your-own sidecars yang menerapkan SDS dan memungkinkan sertifikat dikirim ke Envoy. Mereka termasuk SPIFFE Runtime Environment (SPIRE).

Important

App Mesh tidak menyimpan sertifikat atau kunci pribadi yang digunakan untuk otentikasi TLS bersama. Sebaliknya, Utusan menyimpannya dalam memori.

Konfigurasi titik akhir mesh

Konfigurasi otentikasi TLS timbal balik untuk titik akhir mesh Anda, seperti node virtual atau gateway. Titik akhir ini memberikan sertifikat dan menentukan otoritas terpercaya.

Untuk melakukan ini, Anda perlu menyediakan sertifikat X.509 untuk klien dan server, dan secara eksplisit mendefinisikan sertifikat otoritas terpercaya dalam konteks validasi penghentian TLS dan originasi TLS.

Percaya di dalam jala

Sertifikat sisi server dikonfigurasi dalam pendengar Virtual Node (penghentian TLS), dan sertifikat sisi klien dikonfigurasi dalam backend layanan Virtual Nodes (originasi TLS). Sebagai alternatif untuk konfigurasi ini, Anda dapat menentukan kebijakan klien default untuk semua layanan backend dari node virtual, dan kemudian, jika diperlukan, Anda dapat mengganti kebijakan ini untuk backend tertentu sesuai kebutuhan. Virtual Gateways hanya dapat dikonfigurasi dengan kebijakan klien default yang berlaku untuk semua backend-nya.

Anda dapat mengonfigurasi kepercayaan di berbagai jerat dengan mengaktifkan otentikasi TLS timbal balik untuk lalu lintas masuk di Gateway Virtual untuk kedua jerat.

Percaya di luar jaring

Tentukan sertifikat sisi server di pendengar Virtual Gateway untuk penghentian TLS. Konfigurasi layanan eksternal yang berkomunikasi dengan Virtual Gateway Anda untuk menyajikan sertifikat sisi klien. Sertifikat harus berasal dari salah satu otoritas sertifikat (CAs) yang sama yang digunakan sertifikat sisi server pada pendengar Virtual Gateway untuk originasi TLS.

Migrasikan layanan ke otentikasi TLS bersama

Ikuti panduan ini untuk menjaga konektivitas saat memigrasikan layanan yang ada dalam App Mesh ke autentikasi TLS bersama.

Migrasi layanan yang berkomunikasi melalui plaintext

1. Aktifkan PERMISSIVE mode untuk konfigurasi TLS pada titik akhir server. Mode ini memungkinkan lalu lintas teks biasa untuk terhubung ke titik akhir.
2. Konfigurasi otentikasi TLS timbal balik di server Anda, tentukan sertifikat server, rantai kepercayaan, dan opsional yang terpercaya. SANs
3. Konfirmasikan komunikasi terjadi melalui koneksi TLS.
4. Konfigurasi otentikasi TLS timbal balik pada klien Anda, tentukan sertifikat klien, rantai kepercayaan, dan opsional terpercaya. SANs
5. Aktifkan STRICT mode untuk konfigurasi TLS di server.

Migrasi layanan yang berkomunikasi melalui TLS

1. Konfigurasi pengaturan TLS timbal balik pada klien Anda, tentukan sertifikat klien dan opsional yang tepercaya. SANs Sertifikat klien tidak dikirim ke backend sampai setelah server backend memintanya.
2. Konfigurasi pengaturan TLS timbal balik di server Anda, tentukan rantai kepercayaan dan opsional yang tepercaya. SANs Untuk ini, server Anda meminta sertifikat klien.

Memverifikasi otentikasi TLS timbal balik

Anda dapat merujuk ke dokumentasi [Transport Layer Security: Verify encryption](#) untuk melihat bagaimana tepatnya Envoy memancarkan statistik terkait TLS. Untuk otentikasi TLS timbal balik, Anda harus memeriksa statistik berikut:

- `ssl.handshake`
- `ssl.no_certificate`
- `ssl.fail_verify_no_cert`
- `ssl.fail_verify_san`

Dua contoh statistik berikut bersama-sama menunjukkan bahwa koneksi TLS yang berhasil berakhir ke node virtual semuanya berasal dari klien yang memberikan sertifikat.

```
listener.0.0.0.0_15000.ssl.handshake: 3
```

```
listener.0.0.0.0_15000.ssl.no_certificate: 0
```

Contoh statistik berikutnya menunjukkan bahwa koneksi dari node klien virtual (atau gateway) ke node virtual backend gagal. Nama Alternatif Subjek (SAN) yang disajikan dalam sertifikat server tidak cocok dengan yang SANs dipercaya oleh klien.

```
cluster.cds_egress_my-mesh_my-backend-node_http_9080.ssl.fail_verify_san: 5
```

Panduan otentikasi TLS timbal balik App Mesh

- Panduan [otentikasi TLS Mutual: Panduan](#) ini menjelaskan bagaimana Anda dapat menggunakan App Mesh CLI untuk membangun aplikasi berwarna dengan otentikasi TLS timbal balik.

- Panduan berbasis [Amazon EKS mutual TLS SDS: Panduan ini menunjukkan bagaimana Anda dapat menggunakan otentikasi berbasis TLS SDS](#) timbal balik dengan Amazon EKS dan SPIFFE Runtime Environment (SPIRE).
- Panduan [berbasis file TLS timbal balik Amazon EKS: Panduan ini menunjukkan bagaimana Anda dapat menggunakan otentikasi berbasis file TLS](#) timbal balik dengan Amazon EKS dan SPIFFE Runtime Environment (SPIRE).

Bagaimana AWS App Mesh bekerja dengan IAM

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk AWS App Mesh. Setelah 30 September 2026, Anda tidak lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

AWS Identity and Access Management (IAM) adalah Layanan AWS yang membantu administrator mengontrol akses ke AWS sumber daya dengan aman. Administrator IAM mengontrol siapa yang dapat diautentikasi (masuk) dan diotorisasi (memiliki izin) untuk menggunakan sumber daya App Mesh. IAM adalah Layanan AWS yang dapat Anda gunakan tanpa biaya tambahan.

Topik

- [Audiens](#)
- [Mengautentikasi dengan identitas](#)
- [Mengelola akses menggunakan kebijakan](#)
- [Bagaimana AWS App Mesh bekerja dengan IAM](#)
- [AWS App Mesh contoh kebijakan berbasis identitas](#)
- [AWS kebijakan terkelola untuk App Mesh](#)
- [Menggunakan peran terkait layanan untuk App Mesh](#)
- [Otorisasi Proxy Utusan](#)
- [Memecahkan masalah AWS App Mesh identitas dan akses](#)

Audiens

Cara Anda menggunakan AWS Identity and Access Management (IAM) berbeda berdasarkan peran Anda:

- Pengguna layanan - minta izin dari administrator Anda jika Anda tidak dapat mengakses fitur (lihat [Memecahkan masalah AWS App Mesh identitas dan akses](#))
- Administrator layanan - tentukan akses pengguna dan mengirimkan permintaan izin (lihat [Bagaimana AWS App Mesh bekerja dengan IAM](#))
- Administrator IAM - tulis kebijakan untuk mengelola akses (lihat [AWS App Mesh contoh kebijakan berbasis identitas](#))

Mengautentikasi dengan identitas

Otentikasi adalah cara Anda masuk AWS menggunakan kredensi identitas Anda. Anda harus diautentikasi sebagai Pengguna root akun AWS, pengguna IAM, atau dengan mengasumsikan peran IAM.

Anda dapat masuk sebagai identitas federasi menggunakan kredensial dari sumber identitas seperti AWS IAM Identity Center (Pusat Identitas IAM), autentikasi masuk tunggal, atau kredensial. Google/Facebook Untuk informasi selengkapnya tentang cara masuk, lihat [Cara masuk ke Akun AWS Anda](#) dalam Panduan Pengguna AWS Sign-In .

Untuk akses terprogram, AWS sediakan SDK dan CLI untuk menandatangani permintaan secara kriptografis. Untuk informasi selengkapnya, lihat [AWS Signature Version 4 untuk permintaan API](#) dalam Panduan Pengguna IAM.

Akun AWS pengguna root

Saat Anda membuat Akun AWS, Anda mulai dengan satu identitas masuk yang disebut pengguna Akun AWS root yang memiliki akses lengkap ke semua Layanan AWS dan sumber daya. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari. Untuk tugas yang memerlukan kredensial pengguna root, lihat [Tugas yang memerlukan kredensial pengguna root](#) dalam Panduan Pengguna IAM.

Pengguna dan grup IAM

[Pengguna IAM](#) adalah identitas dengan izin khusus untuk satu orang atau aplikasi. Sebaiknya gunakan kredensial sementara alih-alih pengguna IAM dengan 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.

[Grup IAM](#) menentukan kumpulan pengguna IAM dan mempermudah pengelolaan izin untuk pengguna dalam jumlah besar. Untuk mempelajari selengkapnya, lihat [Kasus penggunaan untuk pengguna IAM](#) dalam Panduan Pengguna IAM.

Peran IAM

[Peran IAM](#) adalah identitas dengan izin khusus yang menyediakan kredensial sementara. Anda dapat mengambil peran dengan [beralih dari pengguna ke peran IAM \(konsol\)](#) atau dengan memanggil operasi AWS CLI atau AWS API. Untuk informasi selengkapnya, lihat [Metode untuk mengambil peran](#) dalam Panduan Pengguna IAM.

Peran IAM berguna untuk akses pengguna terfederasi, izin pengguna IAM sementara, akses lintas akun, akses lintas layanan, dan aplikasi yang berjalan di Amazon EC2. Untuk informasi selengkapnya, lihat [Akses sumber daya lintas akun di IAM](#) dalam Panduan Pengguna IAM.

Mengelola akses menggunakan kebijakan

Anda mengontrol akses AWS dengan membuat kebijakan dan melampirkannya ke AWS identitas atau sumber daya. Kebijakan menentukan izin saat dikaitkan dengan identitas atau sumber daya. AWS mengevaluasi kebijakan ini ketika kepala sekolah membuat permintaan. Sebagian besar kebijakan disimpan AWS sebagai dokumen JSON. Untuk informasi selengkapnya tentang dokumen kebijakan JSON, lihat [Gambaran umum kebijakan JSON](#) dalam Panduan Pengguna IAM.

Menggunakan kebijakan, administrator menentukan siapa yang memiliki akses ke apa dengan mendefinisikan principal mana yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Secara default, pengguna dan peran tidak memiliki izin. Administrator IAM membuat kebijakan IAM dan menambahkannya ke peran, yang kemudian dapat diambil oleh pengguna. Kebijakan IAM mendefinisikan izin terlepas dari metode yang Anda gunakan untuk melakukan operasinya.

Kebijakan berbasis identitas

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang Anda lampirkan ke identitas (pengguna, grup, atau peran). Kebijakan ini mengontrol tindakan apa yang bisa dilakukan oleh identitas tersebut, terhadap sumber daya yang mana, dan dalam kondisi apa. Untuk mempelajari

cara membuat kebijakan berbasis identitas, lihat [Tentukan izin IAM kustom dengan kebijakan yang dikelola pelanggan](#) dalam Panduan Pengguna IAM.

Kebijakan berbasis identitas dapat berupa kebijakan inline (disematkan langsung ke dalam satu identitas) atau kebijakan terkelola (kebijakan mandiri yang dilampirkan pada banyak identitas). Untuk mempelajari cara memilih antara kebijakan terkelola dan kebijakan inline, lihat [Pilih antara kebijakan terkelola 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. Contohnya termasuk 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. Anda harus [menentukan principal](#) dalam kebijakan berbasis sumber daya.

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 dapat menetapkan izin maksimum yang diberikan oleh jenis kebijakan yang lebih umum:

- Batasan izin – Menetapkan izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas kepada entitas IAM. Untuk informasi selengkapnya, lihat [Batasan izin untuk entitas IAM](#) dalam Panduan Pengguna IAM.
- Kebijakan kontrol layanan (SCPs) — Tentukan izin maksimum untuk organisasi atau unit organisasi di AWS Organizations. Untuk informasi selengkapnya, lihat [Kebijakan kontrol layanan](#) dalam Panduan Pengguna AWS Organizations .

- Kebijakan kontrol sumber daya (RCPs) — Tetapkan izin maksimum yang tersedia untuk sumber daya di akun Anda. Untuk informasi selengkapnya, lihat [Kebijakan kontrol sumber daya \(RCPs\)](#) di Panduan AWS Organizations Pengguna.
- Kebijakan sesi – Kebijakan lanjutan yang diteruskan sebagai parameter saat membuat sesi sementara untuk peran atau pengguna terfederasi. 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 AWS App Mesh bekerja dengan IAM

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk AWS App Mesh. Setelah 30 September 2026, Anda tidak lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Sebelum Anda menggunakan IAM untuk mengelola akses ke App Mesh, Anda harus memahami fitur IAM apa yang tersedia untuk digunakan dengan App Mesh. Untuk mendapatkan tampilan tingkat tinggi tentang cara kerja App Mesh dan AWS layanan lainnya dengan IAM, lihat [AWS Layanan yang Bekerja dengan IAM di Panduan Pengguna IAM](#).

Topik

- [Kebijakan berbasis identitas App Mesh](#)
- [Kebijakan berbasis sumber daya App Mesh](#)
- [Otorisasi berdasarkan tag App Mesh](#)
- [Peran IAM App Mesh](#)

Kebijakan berbasis identitas App Mesh

Dengan kebijakan berbasis identitas IAM, Anda dapat menentukan secara spesifik apakah tindakan dan sumber daya diizinkan atau ditolak, serta kondisi yang menjadi dasar dikabulkan atau ditolaknya tindakan tersebut. App Mesh mendukung tindakan, sumber daya, dan kunci kondisi tertentu. Untuk mempelajari semua elemen yang Anda gunakan dalam kebijakan JSON, lihat [Referensi Elemen Kebijakan JSON IAM](#) dalam Panduan Pengguna IAM.

Tindakan

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Yaitu, di mana utama dapat melakukan tindakan pada sumber daya, dan dalam kondisi apa.

Elemen `Action` dari kebijakan JSON menjelaskan tindakan yang dapat Anda gunakan untuk mengizinkan atau menolak akses dalam sebuah kebijakan. Sertakan tindakan dalam kebijakan untuk memberikan izin untuk melakukan operasi terkait.

Tindakan kebijakan di App Mesh menggunakan awalan berikut sebelum tindakan: `appmesh:`. Misalnya, untuk memberikan izin kepada seseorang untuk membuat daftar jerat di akun dengan operasi `appmesh:ListMeshes` API, Anda menyertakan `appmesh:ListMeshes` tindakan tersebut dalam kebijakan mereka. Pernyataan kebijakan harus memuat elemen `Action` atau `NotAction`.

Untuk menentukan beberapa tindakan dalam satu pernyataan, pisahkan tindakan dengan koma seperti berikut:

```
"Action": [  
    "appmesh:ListMeshes",  
    "appmesh:ListVirtualNodes"  
]
```

Anda juga dapat menentukan beberapa tindakan menggunakan wildcard (*). Misalnya, untuk menentukan semua tindakan yang dimulai dengan kata `Describe`, sertakan tindakan berikut.

```
"Action": "appmesh:Describe*"
```

Untuk melihat daftar tindakan App Mesh, lihat [Tindakan yang Ditentukan oleh AWS App Mesh](#) dalam Panduan Pengguna IAM.

Sumber daya

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Yaitu, di mana utama dapat melakukan tindakan pada sumber daya, dan dalam kondisi apa.

Elemen kebijakan JSON `Resource` menentukan objek yang menjadi target penerapan tindakan. Praktik terbaiknya, tentukan sumber daya menggunakan [Amazon Resource Name \(ARN\)](#). Untuk tindakan yang tidak mendukung izin di tingkat sumber daya, gunakan wildcard (*) untuk menunjukkan bahwa pernyataan tersebut berlaku untuk semua sumber daya.

```
"Resource": "*" 
```

meshSumber daya App Mesh memiliki ARN berikut.

```
arn:${Partition}:appmesh:${Region}:${Account}:mesh/${MeshName}
```

Untuk informasi selengkapnya tentang format ARNs, lihat [Amazon Resource Names \(ARNs\) dan Ruang Nama AWS Layanan](#).

Misalnya, untuk menentukan mesh bernama *apps* di *Region-code* Wilayah dalam pernyataan Anda, gunakan ARN berikut.

```
arn:aws:appmesh:Region-code:111122223333:mesh/apps
```

Untuk menentukan semua instance milik akun tertentu, gunakan wildcard (*).

```
"Resource": "arn:aws:appmesh:Region-code:111122223333:mesh/*" 
```

Beberapa tindakan App Mesh, seperti untuk membuat sumber daya, tidak dapat dilakukan pada sumber daya tertentu. Dalam kasus tersebut, Anda harus menggunakan wildcard (*).

```
"Resource": "*" 
```

Banyak tindakan App Mesh API melibatkan banyak sumber daya. Misalnya, `CreateRoute` membuat rute dengan target node virtual, sehingga pengguna IAM harus memiliki izin untuk menggunakan rute

dan node virtual. Untuk menentukan beberapa sumber daya dalam satu pernyataan, pisahkan ARNs dengan koma.

```
"Resource": [  
  "arn:aws:appmesh:Region-code:111122223333:mesh/apps/virtualRouter/serviceB/route/  
  *",  
  "arn:aws:appmesh:Region-code:111122223333:mesh/apps/virtualNode/serviceB"  
]
```

Untuk melihat daftar jenis resource App Mesh dan jenis resource ARNs, lihat [Resources Defined by AWS App Mesh](#) dalam Panduan Pengguna IAM. Untuk mempelajari tindakan mana yang dapat menentukan ARN setiap sumber daya, lihat [Tindakan yang Ditentukan oleh AWS App Mesh](#).

Kunci syarat

App Mesh mendukung penggunaan beberapa tombol kondisi global. Untuk melihat semua kunci syarat global AWS, lihat [Kunci Konteks Syarat Global AWS](#) dalam Panduan Pengguna IAM. Untuk melihat daftar kunci kondisi global yang didukung App Mesh, lihat [Kunci Kondisi untuk AWS App Mesh](#) Panduan Pengguna IAM. Untuk mempelajari tindakan dan sumber daya yang dapat Anda gunakan dengan kunci kondisi, lihat [Tindakan yang Ditentukan oleh AWS App Mesh](#).

Contoh

Untuk melihat contoh kebijakan berbasis identitas App Mesh, lihat [AWS App Mesh contoh kebijakan berbasis identitas](#)

Kebijakan berbasis sumber daya App Mesh

App Mesh tidak mendukung kebijakan berbasis sumber daya. Namun, jika Anda menggunakan layanan AWS Resource Access Manager (AWS RAM) untuk berbagi mesh di seluruh AWS layanan, kebijakan berbasis sumber daya diterapkan ke mesh Anda oleh layanan. AWS RAM Untuk informasi selengkapnya, lihat [Memberikan izin untuk mesh](#).

Otorisasi berdasarkan tag App Mesh

Anda dapat melampirkan tag ke resource App Mesh atau meneruskan tag dalam permintaan ke App Mesh. Untuk mengendalikan akses berdasarkan tanda, berikan informasi tentang tanda di [elemen kondisi](#) dari kebijakan menggunakan kunci kondisi `appmesh:ResourceTag/key-name`, `aws:RequestTag/key-name`, atau `aws:TagKeys`. Untuk informasi selengkapnya tentang menandai resource App Mesh, lihat [Menandai AWS Sumber Daya](#).

Untuk melihat contoh kebijakan berbasis identitas untuk membatasi akses ke sumber daya berdasarkan tag pada sumber daya tersebut, lihat [Membuat jerat App Mesh dengan tag terbatas](#).

Peran IAM App Mesh

[Peran IAM](#) adalah entitas dalam AWS akun Anda yang memiliki izin tertentu.

Menggunakan kredensial sementara dengan App Mesh

Anda dapat menggunakan kredensial sementara untuk masuk dengan gabungan, menjalankan IAM role, atau menjalankan peran lintas akun. Anda memperoleh kredensial keamanan sementara dengan memanggil operasi AWS STS API seperti [AssumeRole](#) atau [GetFederationToken](#).

App Mesh 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 IAM dapat melihat tetapi tidak dapat mengedit izin untuk peran terkait layanan.

App Mesh mendukung peran terkait layanan. Untuk detail tentang membuat atau mengelola peran terkait layanan App Mesh, lihat [Menggunakan peran terkait layanan untuk App Mesh](#).

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, melakukan hal itu dapat merusak fungsionalitas layanan.

App Mesh tidak mendukung peran layanan.

AWS App Mesh contoh kebijakan berbasis identitas

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk. AWS App Mesh Setelah 30 September 2026, Anda tidak lagi dapat

mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Secara default, pengguna dan peran IAM tidak memiliki izin untuk membuat atau memodifikasi resource App Mesh. Mereka juga tidak dapat melakukan tugas menggunakan Konsol Manajemen AWS, 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 pada Tab JSON](#) dalam Panduan Pengguna IAM.

Topik

- [Praktik terbaik kebijakan](#)
- [Menggunakan konsol App Mesh](#)
- [Izinkan para pengguna untuk melihat izin mereka sendiri](#)
- [Buat mesh](#)
- [Buat daftar dan jelaskan semua jerat](#)
- [Membuat jerat App Mesh dengan tag terbatas](#)

Praktik terbaik kebijakan

Kebijakan berbasis identitas menentukan apakah seseorang dapat membuat, mengakses, atau menghapus resource App Mesh di akun Anda. Tindakan ini membuat Akun AWS Anda dikenai biaya. 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 Akun AWS. 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 yang spesifik Layanan AWS, seperti 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 dengan IAM Access Analyzer](#) dalam Panduan Pengguna IAM.
- Memerlukan otentikasi multi-faktor (MFA) - Jika Anda memiliki skenario yang mengharuskan pengguna IAM atau pengguna root di Anda, Akun AWS aktifkan MFA untuk keamanan tambahan. Untuk meminta MFA ketika operasi API dipanggil, tambahkan kondisi MFA pada kebijakan Anda. Untuk informasi selengkapnya, lihat [Amankan akses API dengan MFA](#) dalam Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang praktik terbaik dalam IAM, lihat [Praktik terbaik keamanan di IAM](#) dalam Panduan Pengguna IAM.

Menggunakan konsol App Mesh

Untuk mengakses AWS App Mesh konsol, Anda harus memiliki set izin minimum. Izin ini harus memungkinkan Anda untuk membuat daftar dan melihat detail tentang sumber daya App Mesh di AWS akun Anda. Jika Anda membuat kebijakan berbasis identitas yang lebih ketat daripada izin minimum yang diperlukan, konsol tersebut tidak akan berfungsi sebagaimana mestinya untuk entitas (pengguna IAM atau peran) dengan kebijakan tersebut. Anda dapat melampirkan kebijakan

[AWSAppMeshReadOnly](#) terkelola ke pengguna. Untuk informasi selengkapnya, lihat [Menambahkan Izin ke Pengguna](#) dalam Panduan Pengguna IAM.

Anda tidak perlu mengizinkan izin konsol minimum untuk pengguna yang melakukan panggilan hanya ke AWS CLI atau AWS API. Sebagai alternatif, hanya izinkan akses ke tindakan yang cocok dengan operasi API yang sedang Anda coba lakukan.

Izinkan para pengguna 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 API atau secara terprogram. AWS CLI AWS

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetUserPolicy",
        "iam:ListGroupsWithUser",
        "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 mesh

Contoh ini menunjukkan bagaimana Anda dapat membuat kebijakan yang memungkinkan pengguna membuat mesh untuk akun, di Wilayah mana pun.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "appmesh:CreateMesh",
      "Resource": "arn:aws:appmesh:*:123456789012:CreateMesh"
    }
  ]
}
```

Buat daftar dan jelaskan semua jerat

Contoh ini menunjukkan bagaimana Anda dapat membuat kebijakan yang memungkinkan akses hanya-baca pengguna ke daftar dan menjelaskan semua mesh.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "appmesh:DescribeMesh",
        "appmesh:ListMeshes"
      ]
    }
  ]
}
```

```

    ],
    "Resource": "*"
  }
]
}

```

Membuat jerat App Mesh dengan tag terbatas

Anda dapat menggunakan tag di kebijakan IAM untuk mengontrol tag apa yang bisa diberikan di permintaan IAM. Anda dapat menentukan pasangan nilai kunci tag mana yang dapat ditambahkan, diubah, atau dihapus dari pengguna atau peran IAM. Contoh ini menunjukkan bagaimana Anda dapat membuat kebijakan yang memungkinkan pembuatan mesh, tetapi hanya jika mesh dibuat dengan tag bernama *teamName* dan nilai *booksTeam*.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "appmesh:CreateMesh",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/teamName": "booksTeam"
        }
      }
    }
  ]
}

```

Anda dapat melampirkan kebijakan ini ke pengguna IAM di akun Anda. Jika pengguna mencoba membuat mesh, mesh harus menyertakan tag bernama *teamName* dan nilai *booksTeam*. Jika mesh tidak menyertakan tag dan nilai ini, upaya untuk membuat mesh gagal. Untuk informasi selengkapnya, lihat [Elemen kebijakan IAM JSON: Syarat](#) dalam Panduan Pengguna IAM.

AWS kebijakan terkelola untuk App Mesh

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk AWS App Mesh. Setelah 30 September 2026, Anda tidak akan lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

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, pembaruan akan memengaruhi semua identitas utama (pengguna, grup, dan peran) yang dilampirkan kebijakan tersebut. AWS kemungkinan besar akan memperbarui kebijakan AWS terkelola saat baru Layanan AWS diluncurkan atau operasi API baru tersedia untuk layanan yang ada.

Untuk informasi selengkapnya, lihat [Kebijakan terkelola AWS](#) dalam Panduan Pengguna IAM.

AWS kebijakan terkelola: AWSAppMeshServiceRolePolicy

Anda dapat melampirkan `AWSAppMeshServiceRolePolicy` ke entitas IAM Anda. Memungkinkan akses ke AWS Layanan dan sumber daya yang digunakan atau dikelola oleh AWS App Mesh.

Untuk melihat izin kebijakan ini, lihat [AWSAppMeshServiceRolePolicy](#) di Referensi Kebijakan AWS Terkelola.

Untuk informasi tentang detail izin `AWSAppMeshServiceRolePolicy`, lihat Izin [Peran Tertaut Layanan untuk App Mesh](#).

AWS kebijakan terkelola: AWSApp MeshEnvoyAccess

Anda dapat melampirkan `AWSAppMeshEnvoyAccess` ke entitas IAM Anda. Kebijakan App Mesh Envoy untuk mengakses konfigurasi node virtual.

Untuk melihat izin kebijakan ini, lihat [AWSAppMeshEnvoyAccess](#) di Referensi Kebijakan AWS Terkelola.

AWS kebijakan terkelola: AWSApp MeshFullAccess

Anda dapat melampirkan `AWSAppMeshFullAccess` ke entitas IAM Anda. Menyediakan akses penuh ke AWS App Mesh APIs dan Konsol Manajemen AWS.

Untuk melihat izin kebijakan ini, lihat [AWSAppMeshFullAccess](#) di Referensi Kebijakan AWS Terkelola.

AWS kebijakan terkelola: AWSApp MeshPreviewEnvoyAccess

Anda dapat melampirkan `AWSAppMeshPreviewEnvoyAccess` ke entitas IAM Anda. Kebijakan Utusan Pratinjau App Mesh untuk mengakses konfigurasi node virtual.

Untuk melihat izin kebijakan ini, lihat [AWSAppMeshPreviewEnvoyAccess](#) di Referensi Kebijakan AWS Terkelola.

AWS kebijakan terkelola: AWSApp MeshPreviewServiceRolePolicy

Anda dapat melampirkan `AWSAppMeshPreviewServiceRolePolicy` ke entitas IAM Anda. Memungkinkan akses ke AWS Layanan dan sumber daya yang digunakan atau dikelola oleh AWS App Mesh.

Untuk melihat izin kebijakan ini, lihat [AWSAppMeshPreviewServiceRolePolicy](#) di Referensi Kebijakan AWS Terkelola.

AWS kebijakan terkelola: AWSApp MeshReadOnly

Anda dapat melampirkan `AWSAppMeshReadOnly` ke entitas IAM Anda. Menyediakan akses hanya-baca ke dan. AWS App Mesh APIs Konsol Manajemen AWS

Untuk melihat izin kebijakan ini, lihat [AWSAppMeshReadOnly](#) di Referensi Kebijakan AWS Terkelola.

AWS App Mesh pembaruan kebijakan AWS terkelola

Lihat detail tentang pembaruan kebijakan AWS terkelola AWS App Mesh sejak layanan ini mulai melacak perubahan ini. Untuk peringatan otomatis tentang perubahan pada halaman ini, berlangganan umpan RSS di halaman Riwayat dokumen AWS App Mesh .

Perubahan	Deskripsi	Date
AWSAppMeshFullAccess — Kebijakan yang diperbarui.	Diperbarui AWSAppMeshFullAccess untuk memungkinkan akses ke TagResource dan UntagResource APIs.	April 24, 2024
AWSAppMeshServiceRolePolicy , AWSServiceRoleForAppMesh — Kebijakan yang diperbarui.	Diperbarui AWSServiceRoleForAppMesh dan AWSAppMeshServiceRolePolicy untuk memungkinkan akses ke AWS Cloud Map DiscoverInstancesRevision API.	12 Oktober 2023

Untuk memberikan akses dan menambahkan izin bagi pengguna, grup, atau peran Anda:

- Pengguna dan grup di AWS IAM Identity Center:

Buat rangkaian izin. Ikuti instruksi di [Buat rangkaian izin](#) dalam Panduan Pengguna AWS IAM Identity Center .

- Pengguna yang dikelola di IAM melalui penyedia identitas:

Buat peran untuk federasi identitas. Ikuti instruksi dalam [Buat peran untuk penyedia identitas pihak ketiga \(federasi\)](#) dalam Panduan Pengguna IAM.

- Pengguna IAM:

- Buat peran yang dapat diambil pengguna Anda. Ikuti instruksi dalam [Buat peran untuk pengguna IAM](#) dalam Panduan Pengguna IAM.

- (Tidak disarankan) Lampirkan kebijakan langsung ke pengguna atau tambahkan pengguna ke grup pengguna. Ikuti petunjuk dalam [Menambahkan izin ke pengguna \(konsol\)](#) dalam Panduan Pengguna IAM.

Menggunakan peran terkait layanan untuk App Mesh

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk AWS App Mesh. Setelah 30 September 2026, Anda tidak akan lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

AWS App Mesh menggunakan AWS Identity and Access Management peran [terkait layanan](#) (IAM). Peran terkait layanan adalah jenis unik peran IAM yang ditautkan langsung ke App Mesh. Peran terkait layanan telah ditentukan sebelumnya oleh App Mesh dan menyertakan semua izin yang diperlukan layanan untuk memanggil AWS layanan lain atas nama Anda.

Peran terkait layanan membuat pengaturan App Mesh lebih mudah karena Anda tidak perlu menambahkan izin yang diperlukan secara manual. App Mesh mendefinisikan izin peran terkait layanan, dan kecuali ditentukan lain, hanya App Mesh yang dapat mengambil perannya. Izin-izin yang ditentukan mencakup kebijakan kepercayaan dan kebijakan izin, serta bahwa kebijakan izin tidak dapat dilampirkan ke entitas IAM lainnya.

Anda dapat menghapus peran tertaut layanan hanya setelah terlebih dahulu menghapus sumber dayanya yang terkait. Ini melindungi sumber daya App Mesh karena Anda tidak dapat secara tidak sengaja menghapus izin untuk mengakses sumber daya.

Untuk informasi tentang layanan lain yang mendukung peran terkait layanan, lihat [Layanan AWS yang bisa digunakan dengan IAM](#) dan carilah layanan yang memiliki opsi Ya di kolom Peran Terkait Layanan. Pilih Ya dengan sebuah tautan untuk melihat dokumentasi peran terkait layanan untuk layanan tersebut.

Izin peran terkait layanan untuk App Mesh

App Mesh menggunakan peran terkait layanan bernama `AWSServiceRoleForAppMesh`— Peran ini memungkinkan App Mesh memanggil AWS layanan atas nama Anda.

Peran `AWSServiceRoleForAppMesh` terkait layanan mempercayai `appmesh.amazonaws.com` layanan untuk mengambil peran tersebut.

Detail izin

- `servicediscovery:DiscoverInstances`- Memungkinkan App Mesh untuk menyelesaikan tindakan pada semua AWS sumber daya.
- `servicediscovery:DiscoverInstancesRevision`- Memungkinkan App Mesh untuk menyelesaikan tindakan pada semua AWS sumber daya.

`AWSServiceRoleForAppMesh`

Kebijakan ini mencakup izin berikut:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CloudMapServiceDiscovery",
      "Effect": "Allow",
      "Action": [
        "servicediscovery:DiscoverInstances",
        "servicediscovery:DiscoverInstancesRevision"
      ],
      "Resource": "*"
    },
    {
      "Sid": "ACMCertificateVerification",
      "Effect": "Allow",
      "Action": [
        "acm:DescribeCertificate"
      ],
      "Resource": "*"
    }
  ]
}
```

```
]
}
```

Anda harus mengonfigurasi izin untuk mengizinkan entitas IAM (seperti pengguna, grup, atau peran) untuk membuat, mengedit, atau menghapus peran terkait layanan. Untuk informasi selengkapnya, silakan lihat [Izin Peran Tertaut Layanan](#) di Panduan Pengguna IAM.

Membuat peran terkait layanan untuk App Mesh

Jika Anda membuat mesh setelah 5 Juni 2019 di Konsol Manajemen AWS, the AWS CLI, atau AWS API, App Mesh membuat peran terkait layanan untuk Anda. Agar peran terkait layanan telah dibuat untuk Anda, akun IAM yang Anda gunakan untuk membuat mesh harus memiliki kebijakan [AWSAppMeshFullAccess](#) IAM yang dilampirkan padanya, atau kebijakan yang dilampirkan padanya yang berisi izin. `iam:CreateServiceLinkedRole` 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. Saat Anda membuat mesh, App Mesh membuat peran terkait layanan untuk Anda lagi. Jika akun Anda hanya berisi jerat yang dibuat sebelum 5 Juni 2019 dan Anda ingin menggunakan peran terkait layanan dengan jerat tersebut, maka Anda dapat membuat peran menggunakan konsol IAM.

Anda dapat menggunakan konsol IAM untuk membuat peran terkait layanan dengan kasus penggunaan App Mesh. Di AWS CLI atau AWS API, buat peran terkait layanan dengan nama `appmesh.amazonaws.com` layanan. Untuk informasi lebih lanjut, lihat [Membuat Peran yang Terhubung dengan Layanan](#) di Panduan Pengguna IAM. Jika Anda menghapus peran tertaut layanan ini, Anda dapat mengulang proses yang sama untuk membuat peran tersebut lagi.

Mengedit peran terkait layanan untuk App Mesh

App Mesh tidak memungkinkan Anda mengedit peran `AWSServiceRoleForAppMesh` terkait layanan. Setelah membuat peran terkait layanan, Anda tidak dapat mengubah nama peran karena berbagai entitas mungkin merujuk peran tersebut. Namun, Anda dapat mengedit penjelasan peran menggunakan IAM. Untuk informasi selengkapnya, lihat [Mengedit Peran Tertaut Layanan](#) dalam Panduan Pengguna IAM.

Menghapus peran terkait layanan untuk App Mesh

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

yang tidak dipantau atau dipelihara secara aktif. Tetapi, Anda harus membersihkan sumber daya peran yang terhubung dengan layanan sebelum menghapusnya secara manual.

Note

Jika layanan App Mesh menggunakan peran saat Anda mencoba menghapus sumber daya, penghapusan mungkin gagal. Jika hal itu terjadi, tunggu beberapa menit dan coba mengoperasikannya lagi.

Untuk menghapus resource App Mesh yang digunakan oleh AWSService RoleForAppMesh

1. Hapus semua [route](#) yang ditentukan untuk semua router di mesh.
2. Hapus semua [router virtual](#) di mesh.
3. Hapus semua [layanan virtual](#) di mesh.
4. Hapus semua [node virtual](#) di mesh.
5. Hapus [jala](#).

Selesaikan langkah-langkah sebelumnya untuk semua jerat di akun Anda.

Untuk menghapus peran tertaut layanan secara manual menggunakan IAM

Gunakan konsol IAM, the AWS CLI, atau AWS API untuk menghapus peran AWSService RoleForAppMesh terkait layanan. Untuk informasi selengkapnya, silakan lihat [Menghapus Peran Terkait Layanan](#) di Panduan Pengguna IAM.

Wilayah yang Didukung untuk peran terkait layanan App Mesh

App Mesh mendukung penggunaan peran terkait layanan di semua Wilayah tempat layanan tersedia. Untuk informasi selengkapnya, lihat [Titik Akhir dan Kuota App Mesh](#).

Otorisasi Proxy Utusan

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk. AWS App Mesh Setelah 30 September 2026, Anda tidak lagi dapat

mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Otorisasi proxy mengotorisasi proxy [Envoy](#) yang berjalan dalam tugas Amazon ECS, di pod Kubernetes yang berjalan di Amazon EKS, atau berjalan pada instance Amazon EC2 untuk membaca konfigurasi satu atau beberapa titik akhir mesh dari Layanan Manajemen Utusan App Mesh. Untuk akun pelanggan yang sudah memiliki Utusan yang terhubung ke titik akhir App Mesh mereka sebelum 26/04/2021, otorisasi proxy diperlukan untuk node virtual yang menggunakan [Transport Layer Security \(TLS\) dan untuk gateway virtual \(dengan atau tanpa TLS\)](#). Untuk akun pelanggan yang ingin menghubungkan Utusan ke titik akhir App Mesh mereka setelah 26/04/2021, otorisasi proxy diperlukan untuk semua kemampuan App Mesh. Disarankan untuk semua akun pelanggan untuk mengaktifkan otorisasi proxy untuk semua node virtual, bahkan jika mereka tidak menggunakan TLS, untuk memiliki pengalaman yang aman dan konsisten menggunakan IAM untuk otorisasi ke sumber daya tertentu. Otorisasi proxy mengharuskan `appmesh:StreamAggregatedResources` izin ditentukan dalam kebijakan IAM. Kebijakan harus dilampirkan ke peran IAM, dan peran IAM tersebut harus dilampirkan ke sumber daya komputasi tempat Anda meng-host proxy.

Buat kebijakan IAM

Jika Anda ingin semua titik akhir mesh di mesh layanan dapat membaca konfigurasi untuk semua titik akhir mesh, lewati ke [Buat peran IAM](#). Jika Anda ingin membatasi titik akhir mesh yang konfigurasi dapat dibaca oleh titik akhir mesh individual, maka Anda perlu membuat satu atau lebih kebijakan IAM. Membatasi titik akhir mesh yang konfigurasi dapat dibaca dari hanya proxy Envoy yang berjalan pada sumber daya komputasi tertentu direkomendasikan. Buat kebijakan IAM dan tambahkan `appmesh:StreamAggregatedResources` izin ke kebijakan. Contoh kebijakan berikut memungkinkan konfigurasi node virtual bernama `serviceBv1` dan `serviceBv2` dibaca dalam mesh layanan. Konfigurasi tidak dapat dibaca untuk node virtual lain yang ditentukan dalam mesh layanan. Untuk informasi selengkapnya tentang membuat atau mengedit kebijakan IAM, lihat [Membuat Kebijakan IAM dan Mengedit Kebijakan IAM](#).

JSON

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

```
{
  "Effect": "Allow",
  "Action": "appmesh:StreamAggregatedResources",
  "Resource": [
    "arn:aws:appmesh:us-east-1:123456789012:mesh/app1/virtualNode/
serviceBv1",
    "arn:aws:appmesh:us-east-1:123456789012:mesh/app1/virtualNode/
serviceBv2"
  ]
}
```

Anda dapat membuat beberapa kebijakan, dengan setiap kebijakan membatasi akses ke titik akhir mesh yang berbeda.

Buat peran IAM

Jika Anda ingin semua titik akhir mesh di mesh layanan dapat membaca konfigurasi untuk semua titik akhir mesh, maka Anda hanya perlu membuat satu peran IAM. Jika Anda ingin membatasi titik akhir mesh tempat konfigurasi dapat dibaca oleh titik akhir mesh individual, maka Anda perlu membuat peran untuk setiap kebijakan yang Anda buat pada langkah sebelumnya. Lengkapi instruksi untuk sumber daya komputasi yang dijalankan proxy.

- Amazon EKS — Jika Anda ingin menggunakan peran singe, Anda dapat menggunakan peran yang ada yang dibuat dan ditetapkan ke node pekerja saat membuat kluster. Untuk menggunakan beberapa peran, kluster Anda harus memenuhi persyaratan yang ditentukan dalam [Mengaktifkan Peran IAM untuk Akun Layanan di Cluster Anda](#). Buat peran IAM dan kaitkan peran dengan akun layanan Kubernetes. Untuk informasi selengkapnya, lihat [Membuat Peran dan Kebijakan IAM untuk Akun Layanan Anda](#) dan [Menentukan Peran IAM untuk Akun Layanan Anda](#).
- Amazon ECS — Pilih AWS layanan, pilih Layanan Kontainer Elastis, lalu pilih kasus penggunaan Tugas Layanan Kontainer Elastis saat membuat peran IAM Anda.
- Amazon EC2 — Pilih AWS layanan, pilih EC2, lalu pilih kasus penggunaan EC2 saat membuat peran IAM Anda. Ini berlaku baik Anda meng-host proxy secara langsung di instans Amazon EC2 atau di Kubernetes yang berjalan pada sebuah instance.

Untuk informasi selengkapnya tentang cara membuat peran IAM, lihat [Membuat Peran untuk AWS Layanan](#).

Lampirkan kebijakan IAM

Jika Anda ingin semua titik akhir mesh di mesh layanan dapat membaca konfigurasi untuk semua titik akhir mesh, lampirkan kebijakan IAM [AWSAppMeshEnvoyAccess](#) terkelola ke peran IAM yang Anda buat pada langkah sebelumnya. Jika Anda ingin membatasi titik akhir mesh tempat konfigurasi dapat dibaca oleh titik akhir mesh individual, lampirkan setiap kebijakan yang Anda buat ke setiap peran yang Anda buat. Untuk informasi selengkapnya tentang melampirkan kebijakan IAM kustom atau terkelola ke peran IAM, lihat [Menambahkan Izin Identitas IAM](#).

Lampirkan peran IAM

Lampirkan setiap peran IAM ke sumber daya komputasi yang sesuai:

- Amazon EKS — Jika Anda melampirkan kebijakan ke peran yang dilampirkan ke node pekerja Anda, Anda dapat melewati langkah ini. Jika Anda membuat peran terpisah, tetapkan setiap peran ke akun layanan Kubernetes yang terpisah, dan tetapkan setiap akun layanan ke spesifikasi penerapan pod Kubernetes individual yang menyertakan proxy Envoy. Untuk informasi selengkapnya, lihat [Menentukan Peran IAM untuk Akun Layanan Anda](#) di Panduan Pengguna Amazon EKS dan [Mengonfigurasi Akun Layanan untuk Pod](#) dalam dokumentasi Kubernetes.
- Amazon ECS — Lampirkan Peran Tugas Amazon ECS ke definisi tugas yang menyertakan proxy Utusan. Tugas dapat digunakan dengan tipe peluncuran EC2 atau Fargate. Untuk informasi selengkapnya tentang cara membuat Peran Tugas Amazon ECS dan melampirkannya ke tugas, lihat [Menentukan Peran IAM untuk Tugas Anda](#).
- Amazon EC2 — Peran IAM harus dilampirkan ke instans Amazon EC2 yang menghosting proxy Envoy. Untuk informasi selengkapnya tentang cara melampirkan peran ke instans Amazon EC2, lihat [Saya telah membuat peran IAM, dan sekarang saya ingin menentukannya ke instans EC2](#).

Konfirmasikan izin

Konfirmasikan bahwa `appmesh:StreamAggregatedResources` izin ditetapkan ke sumber daya komputasi tempat Anda meng-host proxy dengan memilih salah satu nama layanan komputasi.

Amazon EKS

Kebijakan kustom dapat ditetapkan ke peran yang ditetapkan ke node pekerja, ke masing-masing pod, atau keduanya. Namun, disarankan agar Anda menetapkan kebijakan hanya pada masing-masing pod, sehingga Anda dapat membatasi akses pod individual ke titik akhir mesh individual. Jika kebijakan dilampirkan ke peran yang ditetapkan ke node pekerja, pilih tab Amazon EC2, dan

selesaikan langkah-langkah yang ditemukan di sana untuk instance node pekerja Anda. Untuk menentukan peran IAM mana yang ditetapkan ke pod Kubernetes, selesaikan langkah-langkah berikut.

1. Lihat detail penerapan Kubernetes yang menyertakan pod yang ingin Anda konfirmasi bahwa akun layanan Kubernetes telah ditetapkan. Perintah berikut menampilkan detail untuk penyebaran bernama *my-deployment*.

```
kubectl describe deployment my-deployment
```

Dalam output yang dikembalikan perhatikan nilai di sebelah kanan `Service Account`:. Jika baris yang dimulai dengan `Service Account`: tidak ada, maka akun layanan Kubernetes kustom saat ini tidak ditetapkan ke penerapan. Anda harus menetapkan satu. Untuk informasi selengkapnya, lihat [Mengatur Service Account untuk Pod](#) dalam dokumentasi Kubernetes.

2. Lihat detail akun layanan yang dikembalikan pada langkah sebelumnya. Perintah berikut menampilkan rincian akun layanan bernama *my-service-account*.

```
kubectl describe serviceaccount my-service-account
```

Asalkan akun layanan Kubernetes dikaitkan dengan AWS Identity and Access Management peran, salah satu baris yang dikembalikan akan terlihat mirip dengan contoh berikut.

```
Annotations:          eks.amazonaws.com/role-arn=arn:aws:iam::123456789012:role/  
my-deployment
```

Pada contoh sebelumnya *my-deployment* adalah nama peran IAM yang dikaitkan dengan akun layanan. Jika output akun layanan tidak berisi baris yang mirip dengan contoh di atas, maka akun layanan Kubernetes tidak terkait dengan AWS Identity and Access Management akun dan Anda perlu mengaitkannya ke akun. Untuk informasi selengkapnya, lihat [Menentukan Peran IAM untuk Akun Layanan Anda](#).

3. Masuk ke Konsol Manajemen AWS dan buka konsol IAM di <https://console.aws.amazon.com/iam/>.
4. Di navigasi kiri, pilih Peran. Pilih nama peran IAM yang Anda catat di langkah sebelumnya.
5. Konfirmasikan bahwa kebijakan kustom yang Anda buat sebelumnya, atau kebijakan [AWSAppMeshEnvoyAccess](#) terkelola tercantum. Jika tidak ada kebijakan yang dilampirkan, [lampirkan kebijakan IAM](#) ke peran IAM. Jika Anda ingin melampirkan kebijakan IAM khusus

tetapi tidak memilikinya, maka Anda perlu [membuat kebijakan IAM khusus dengan izin](#) yang diperlukan. Jika kebijakan IAM khusus dilampirkan, pilih kebijakan tersebut dan konfirmasi bahwa kebijakan tersebut berisi "Action": "appmesh:StreamAggregatedResources". Jika tidak, maka Anda perlu menambahkan izin itu ke kebijakan IAM khusus Anda. Anda juga dapat mengonfirmasi bahwa Nama Sumber Daya Amazon (ARN) yang sesuai untuk titik akhir mesh tertentu terdaftar. Jika tidak ARNs terdaftar, Anda dapat mengedit kebijakan untuk menambah, menghapus, atau mengubah daftar ARNs. Untuk informasi selengkapnya, lihat [Mengedit Kebijakan IAM](#) dan [Buat kebijakan IAM](#).

6. Ulangi langkah sebelumnya untuk setiap pod Kubernetes yang berisi proxy Envoy.

Amazon ECS

1. Dari konsol Amazon ECS, pilih Definisi Tugas.
2. Pilih tugas Amazon ECS Anda.
3. Pada halaman Nama Definisi Tugas, pilih definisi tugas Anda.
4. Pada halaman Definisi Tugas, pilih tautan nama peran IAM yang ada di sebelah kanan Peran Tugas. Jika peran IAM tidak terdaftar, maka Anda perlu [membuat peran IAM](#) dan melampirkannya ke tugas Anda dengan [memperbarui definisi tugas Anda](#).
5. Di halaman Ringkasan, pada tab Izin, konfirmasi bahwa kebijakan kustom yang Anda buat sebelumnya, atau kebijakan [AWSAppMeshEnvoyAccess](#) terkelola tercantum. Jika tidak ada kebijakan yang dilampirkan, [lampirkan kebijakan IAM](#) ke peran IAM. Jika Anda ingin melampirkan kebijakan IAM khusus tetapi tidak memilikinya, maka Anda perlu [membuat kebijakan IAM kustom](#). Jika kebijakan IAM khusus dilampirkan, pilih kebijakan tersebut dan konfirmasi bahwa kebijakan tersebut berisi "Action": "appmesh:StreamAggregatedResources". Jika tidak, maka Anda perlu menambahkan izin itu ke kebijakan IAM khusus Anda. Anda juga dapat mengonfirmasi bahwa Nama Sumber Daya Amazon (ARN) yang sesuai untuk titik akhir mesh tertentu terdaftar. Jika tidak ARNs terdaftar, Anda dapat mengedit kebijakan untuk menambah, menghapus, atau mengubah daftar ARNs. Untuk informasi selengkapnya, lihat [Mengedit Kebijakan IAM](#) dan [Buat kebijakan IAM](#).
6. Ulangi langkah sebelumnya untuk setiap definisi tugas yang berisi proxy Utusan.

Amazon EC2

1. Dari konsol Amazon EC2, pilih Instans di navigasi kiri.

2. Pilih salah satu instance Anda yang menghosting proxy Utusan.
3. Di tab Deskripsi, pilih tautan nama peran IAM yang ada di sebelah kanan peran IAM. Jika peran IAM tidak terdaftar, maka Anda perlu [membuat peran IAM](#).
4. Di halaman Ringkasan, pada tab Izin, konfirmasi bahwa kebijakan kustom yang Anda buat sebelumnya, atau kebijakan [AWSAppMeshEnvoyAccess](#) terkelola tercantum. Jika tidak ada kebijakan yang dilampirkan, [lampirkan kebijakan IAM](#) ke peran IAM. Jika Anda ingin melampirkan kebijakan IAM khusus tetapi tidak memilikinya, maka Anda perlu [membuat kebijakan IAM kustom](#). Jika kebijakan IAM khusus dilampirkan, pilih kebijakan tersebut dan konfirmasi bahwa kebijakan tersebut berisi "Action" : "appmesh:StreamAggregatedResources". Jika tidak, maka Anda perlu menambahkan izin itu ke kebijakan IAM khusus Anda. Anda juga dapat mengonfirmasi bahwa Nama Sumber Daya Amazon (ARN) yang sesuai untuk titik akhir mesh tertentu terdaftar. Jika tidak ARNs terdaftar, Anda dapat mengedit kebijakan untuk menambah, menghapus, atau mengubah daftar ARNs. Untuk informasi selengkapnya, lihat [Mengedit Kebijakan IAM](#) dan [Buat kebijakan IAM](#).
5. Ulangi langkah sebelumnya untuk setiap instance tempat Anda meng-host proxy Envoy.

Memecahkan masalah AWS App Mesh identitas dan akses

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk AWS App Mesh. Setelah 30 September 2026, Anda tidak lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang mungkin Anda temui saat bekerja dengan App Mesh dan IAM.

Topik

- [Saya tidak berwenang untuk melakukan tindakan di App Mesh](#)
- [Saya ingin mengizinkan orang di luar AWS akun saya untuk mengakses sumber daya App Mesh saya](#)

Saya tidak berwenang untuk melakukan tindakan di App Mesh

Jika Konsol Manajemen AWS memberitahu Anda bahwa Anda tidak berwenang untuk melakukan tindakan, maka Anda harus menghubungi administrator Anda untuk bantuan. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Kesalahan berikut terjadi ketika pengguna mateojackson IAM mencoba menggunakan konsol untuk membuat node virtual bernama *my-virtual-node* dalam mesh bernama *my-mesh* tetapi tidak memiliki `appmesh:CreateVirtualNode` izin.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to
perform: appmesh:CreateVirtualNode on resource: arn:aws:appmesh:us-
east-1:123456789012:mesh/my-mesh/virtualNode/my-virtual-node
```

Dalam hal ini, Mateo meminta administratornya untuk memperbarui kebijakannya untuk memungkinkannya membuat simpul virtual menggunakan `appmesh:CreateVirtualNode` tindakan tersebut.

Note

Karena node virtual dibuat dalam mesh, akun Mateo juga memerlukan `appmesh:DescribeMesh` dan `appmesh:ListMeshes` tindakan untuk membuat node virtual di konsol.

Saya ingin mengizinkan orang di luar AWS akun saya untuk mengakses sumber daya App Mesh 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, periksa referensi berikut:

- Untuk mengetahui apakah App Mesh mendukung fitur ini, lihat [Bagaimana AWS App Mesh bekerja dengan IAM](#).

- Untuk mempelajari cara menyediakan akses ke sumber daya Anda di seluruh sumber daya Akun AWS yang Anda miliki, lihat [Menyediakan akses ke pengguna IAM di pengguna lain Akun AWS yang Anda miliki](#) di Panduan Pengguna IAM.
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda kepada pihak ketiga Akun AWS, lihat [Menyediakan akses yang Akun AWS dimiliki oleh pihak ketiga](#) dalam 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 [Akses sumber daya lintas akun di IAM di Panduan Pengguna IAM](#).

Logging panggilan AWS App Mesh API menggunakan AWS CloudTrail

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk AWS App Mesh. Setelah 30 September 2026, Anda tidak akan lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

AWS App Mesh terintegrasi dengan [AWS CloudTrail](#), layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau Layanan AWS. CloudTrail menangkap semua panggilan API untuk App Mesh sebagai peristiwa. Panggilan yang diambil mencakup panggilan dari konsol App Mesh dan panggilan kode ke operasi App Mesh API. Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat ke App Mesh, alamat IP dari mana permintaan dibuat, kapan dibuat, dan detail tambahan.

Setiap entri peristiwa atau log berisi informasi tentang entitas yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan berikut hal ini:

- Baik permintaan tersebut dibuat dengan kredensial pengguna root atau pengguna.
- Apakah permintaan dibuat atas nama pengguna IAM Identity Center.

- Apakah permintaan tersebut dibuat dengan kredensial keamanan sementara untuk satu peran atau pengguna gabungan.
- Apakah permintaan tersebut dibuat oleh Layanan AWS lain.

CloudTrail aktif di Anda Akun AWS ketika Anda membuat akun dan Anda secara otomatis memiliki akses ke riwayat CloudTrail Acara. Riwayat CloudTrail Acara menyediakan catatan yang dapat dilihat, dapat dicari, dapat diunduh, dan tidak dapat diubah dari 90 hari terakhir dari peristiwa manajemen yang direkam dalam file. Wilayah AWS Untuk informasi selengkapnya, lihat [Bekerja dengan riwayat CloudTrail Acara](#) di Panduan AWS CloudTrail Pengguna. Tidak ada CloudTrail biaya untuk melihat riwayat Acara.

Untuk catatan acara yang sedang berlangsung dalam 90 hari Akun AWS terakhir Anda, buat jejak atau penyimpanan data acara [CloudTrailDanau](#).

CloudTrail jalan setapak

Jejak memungkinkan CloudTrail untuk mengirimkan file log ke bucket Amazon S3. Semua jalur yang dibuat menggunakan Konsol Manajemen AWS Multi-region. Anda dapat membuat jalur Single-region atau Multi-region dengan menggunakan AWS CLI Membuat jejak Multi-wilayah disarankan karena Anda menangkap aktivitas Wilayah AWS di semua akun Anda. Jika Anda membuat jejak wilayah Tunggal, Anda hanya dapat melihat peristiwa yang dicatat di jejak. Wilayah AWS Untuk informasi selengkapnya tentang jejak, lihat [Membuat jejak untuk Anda Akun AWS](#) dan [Membuat jejak untuk organisasi](#) di Panduan AWS CloudTrail Pengguna.

Anda dapat mengirimkan satu salinan acara manajemen yang sedang berlangsung ke bucket Amazon S3 Anda tanpa biaya CloudTrail dengan membuat jejak, namun, ada biaya penyimpanan Amazon S3. Untuk informasi selengkapnya tentang CloudTrail harga, lihat [AWS CloudTrail Harga](#). Untuk informasi tentang harga Amazon S3, lihat [Harga Amazon S3](#).

CloudTrail Menyimpan data acara danau

CloudTrail Lake memungkinkan Anda menjalankan kueri berbasis SQL pada acara Anda. CloudTrail [Lake mengonversi peristiwa yang ada dalam format JSON berbasis baris ke format Apache ORC](#). ORC adalah format penyimpanan kolumnar yang dioptimalkan untuk pengambilan data dengan cepat. Peristiwa digabungkan ke dalam penyimpanan data peristiwa, yang merupakan kumpulan peristiwa yang tidak dapat diubah berdasarkan kriteria yang Anda pilih dengan menerapkan pemilih acara [tingkat lanjut](#). Penyeleksi yang Anda terapkan ke penyimpanan data acara mengontrol peristiwa mana yang bertahan dan tersedia untuk Anda

kueri. Untuk informasi lebih lanjut tentang CloudTrail Danau, lihat [Bekerja dengan AWS CloudTrail Danau](#) di Panduan AWS CloudTrail Pengguna.

CloudTrail Penyimpanan data acara danau dan kueri menimbulkan biaya. Saat Anda membuat penyimpanan data acara, Anda memilih [opsi harga](#) yang ingin Anda gunakan untuk penyimpanan data acara. Opsi penetapan harga menentukan biaya untuk menelan dan menyimpan peristiwa, dan periode retensi default dan maksimum untuk penyimpanan data acara. Untuk informasi selengkapnya tentang CloudTrail harga, lihat [AWS CloudTrail Harga](#).

Acara manajemen App Mesh di CloudTrail

[Acara manajemen](#) memberikan informasi tentang operasi manajemen yang dilakukan pada sumber daya di Anda Akun AWS. Ini juga dikenal sebagai operasi bidang kontrol. Secara default, CloudTrail mencatat peristiwa manajemen.

AWS App Mesh mencatat semua operasi bidang kontrol App Mesh sebagai peristiwa manajemen. Untuk daftar operasi bidang AWS App Mesh kontrol tempat App Mesh log ke log CloudTrail, lihat [Referensi AWS App Mesh API](#).

Contoh acara App Mesh

Peristiwa mewakili permintaan tunggal dari sumber manapun dan mencakup informasi tentang operasi API yang diminta, tanggal dan waktu operasi, parameter permintaan, dan sebagainya. CloudTrail file log bukanlah jejak tumpukan yang diurutkan dari panggilan API publik, sehingga peristiwa tidak muncul dalam urutan tertentu.

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan `StreamAggregatedResources` tindakan.

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAIOSFODNN7EXAMPLE:d060be4ac3244e05aca4e067bfe241f8",
    "arn": "arn:aws:sts::123456789012:assumed-role/Application-TaskIamRole-C20GBLBRLBXE/d060be4ac3244e05aca4e067bfe241f8",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "invokedBy": "appmesh.amazonaws.com"
  },
```

```
"eventTime": "2021-06-09T23:09:46Z",
"eventSource": "appmesh.amazonaws.com",
"eventName": "StreamAggregatedResources",
"awsRegion": "us-west-2",
"sourceIPAddress": "appmesh.amazonaws.com",
"userAgent": "appmesh.amazonaws.com",
"eventID": "e3c6f4ce-EXAMPLE",
"readOnly": false,
"eventType": "AwsServiceEvent",
"managementEvent": true,
"recipientAccountId": "123456789012",
"serviceEventDetails": {
  "connectionId": "e3c6f4ce-EXAMPLE",
  "nodeArn": "arn:aws:appmesh:us-west-2:123456789012:mesh/CloudTrail-Test/
virtualNode/cloudtrail-test-vn",
  "eventStatus": "ConnectionEstablished",
  "failureReason": ""
},
"eventCategory": "Management"
}
```

Untuk informasi tentang konten CloudTrail rekaman, lihat [konten CloudTrail rekaman](#) di Panduan AWS CloudTrail Pengguna.

Perlindungan data di AWS App Mesh

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk. AWS App Mesh Setelah 30 September 2026, Anda tidak lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

[Model tanggung jawab AWS bersama model](#) berlaku untuk perlindungan data di AWS App Mesh. Seperti yang dijelaskan dalam model AWS ini, bertanggung jawab untuk melindungi infrastruktur global yang menjalankan semua AWS Cloud. Anda bertanggung jawab untuk mempertahankan kendali atas konten yang di-host pada infrastruktur ini. Anda juga bertanggung jawab atas tugas-tugas konfigurasi dan manajemen keamanan untuk Layanan AWS yang Anda gunakan. Lihat

informasi yang lebih lengkap tentang privasi data dalam [Pertanyaan Umum Privasi Data](#). Lihat informasi tentang perlindungan data di Eropa di pos blog [Model Tanggung Jawab Bersama dan GDPR AWS](#) di Blog Keamanan AWS .

Untuk tujuan perlindungan data, kami menyarankan Anda melindungi Akun AWS kredensial dan mengatur pengguna individu dengan AWS IAM Identity Center atau AWS Identity and Access Management (IAM). Dengan cara itu, setiap pengguna hanya diberi izin yang diperlukan untuk memenuhi tanggung jawab tugasnya. Kami juga menyarankan supaya Anda mengamankan data dengan cara-cara berikut:

- Gunakan autentikasi multi-faktor (MFA) pada setiap akun.
- Gunakan SSL/TLS untuk berkomunikasi dengan sumber daya. AWS Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Siapkan API dan pencatatan aktivitas pengguna dengan AWS CloudTrail. Untuk informasi tentang penggunaan CloudTrail jejak untuk menangkap AWS aktivitas, lihat [Bekerja dengan CloudTrail jejak](#) di AWS CloudTrail Panduan Pengguna.
- Gunakan solusi AWS enkripsi, bersama dengan semua kontrol keamanan default di dalamnya Layanan AWS.
- Gunakan layanan keamanan terkelola tingkat lanjut seperti Amazon Macie, yang membantu menemukan dan mengamankan data sensitif yang disimpan di Amazon S3.
- Jika Anda memerlukan modul kriptografi tervalidasi FIPS 140-3 saat mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Lihat informasi selengkapnya tentang titik akhir FIPS yang tersedia di [Standar Pemrosesan Informasi Federal \(FIPS\) 140-3](#).

Kami sangat merekomendasikan agar Anda tidak pernah memasukkan informasi identifikasi yang sensitif, seperti nomor rekening pelanggan Anda, ke dalam tanda atau bidang isian bebas seperti bidang Nama. Ini termasuk saat Anda bekerja dengan App Mesh atau lainnya Layanan AWS menggunakan konsol, API AWS CLI, atau AWS SDKs. Data apa pun yang Anda masukkan ke dalam tanda atau bidang isian bebas yang digunakan untuk nama dapat digunakan untuk log penagihan atau log diagnostik. Saat Anda memberikan URL ke server eksternal, kami sangat menganjurkan supaya Anda tidak menyertakan informasi kredensial di dalam URL untuk memvalidasi permintaan Anda ke server itu.

Enkripsi data

Data Anda dienkripsi saat menggunakan App Mesh.

Enkripsi diam

Secara default, konfigurasi App Mesh yang Anda buat dienkripsi saat istirahat.

Enkripsi bergerak

Titik akhir layanan App Mesh menggunakan protokol HTTPS. Semua komunikasi antara proxy Utusan dan Layanan Manajemen Utusan App Mesh dienkripsi. Jika Anda memerlukan enkripsi yang sesuai dengan FIPS untuk komunikasi antara proxy Utusan dan Layanan Manajemen Utusan App Mesh, ada varian FIPS dari gambar wadah proxy Envoy yang dapat Anda gunakan. Untuk informasi selengkapnya, lihat [Gambar utusan](#).

Komunikasi antar kontainer dalam node virtual tidak dienkripsi, tetapi lalu lintas ini tidak meninggalkan namespace jaringan.

Validasi kepatuhan untuk AWS App Mesh

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk. AWS App Mesh Setelah 30 September 2026, Anda tidak lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Untuk mempelajari apakah an Layanan AWS berada dalam lingkup program kepatuhan tertentu, lihat [Layanan AWS di Lingkup oleh Program Kepatuhan Layanan AWS](#) dan pilih program kepatuhan yang Anda minati. Untuk informasi umum, lihat [Program AWS Kepatuhan Program AWS](#) .

Anda dapat mengunduh laporan audit pihak ketiga menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat [Mengunduh Laporan di AWS Artifact](#) .

Tanggung jawab kepatuhan Anda saat menggunakan Layanan AWS ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, dan hukum dan peraturan yang berlaku. Untuk informasi selengkapnya tentang tanggung jawab kepatuhan Anda saat menggunakan Layanan AWS, lihat [Dokumentasi AWS Keamanan](#).

Keamanan infrastruktur di AWS App Mesh

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk. AWS App Mesh Setelah 30 September 2026, Anda tidak akan lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Sebagai layanan terkelola, AWS App Mesh dilindungi oleh keamanan jaringan AWS global. 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 App Mesh 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.

Anda dapat meningkatkan postur keamanan VPC Anda dengan mengonfigurasi App Mesh untuk menggunakan titik akhir VPC antarmuka. Lihat informasi yang lebih lengkap di [Antarmuka App Mesh titik akhir VPC \(AWS PrivateLink\)](#).

Antarmuka App Mesh titik akhir VPC (AWS PrivateLink)

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk. AWS App Mesh Setelah 30 September 2026, Anda tidak akan lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi

lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Anda dapat meningkatkan postur keamanan VPC Amazon Anda dengan mengonfigurasi App Mesh untuk menggunakan titik akhir VPC antarmuka. Endpoint antarmuka didukung oleh AWS PrivateLink, teknologi yang memungkinkan Anda mengakses App Mesh secara pribadi APIs dengan menggunakan alamat IP pribadi. PrivateLink membatasi semua lalu lintas jaringan antara VPC Amazon Anda dan App Mesh ke jaringan Amazon.

Anda tidak diharuskan untuk mengkonfigurasi PrivateLink, tetapi kami merekomendasikannya. Untuk informasi selengkapnya tentang PrivateLink dan antarmuka titik akhir VPC, lihat [Mengakses Layanan Melalui AWS PrivateLink](#)

Pertimbangan untuk titik akhir VPC antarmuka App Mesh

Sebelum Anda mengatur titik akhir VPC antarmuka untuk App Mesh, perhatikan pertimbangan berikut:

- Jika VPC Amazon Anda tidak memiliki gateway internet dan tugas Anda menggunakan driver `awslogs` log untuk mengirim informasi log ke Log, Anda harus membuat antarmuka VPC endpoint untuk CloudWatch Log. CloudWatch Untuk informasi selengkapnya, lihat [Menggunakan CloudWatch Log dengan Titik Akhir VPC Antarmuka di Panduan Pengguna Amazon CloudWatch Logs](#).
- Titik akhir VPC tidak mendukung AWS permintaan lintas wilayah. Pastikan Anda membuat titik akhir di Wilayah yang sama tempat Anda berencana mengeluarkan panggilan API ke App Mesh.
- Titik akhir VPC hanya mendukung DNS yang disediakan Amazon melalui Amazon Route 53. Jika Anda ingin menggunakan DNS Anda sendiri, Anda dapat menggunakan penerusan DNS bersyarat. Untuk informasi selengkapnya, lihat [Pengaturan DHCP](#) dalam Panduan Pengguna Amazon VPC.
- Grup keamanan yang terpasang pada titik akhir VPC harus mengizinkan koneksi masuk pada port 443 dari subnet pribadi VPC Amazon.

Note

Mengontrol akses ke App Mesh dengan melampirkan kebijakan titik akhir ke titik akhir VPC (misalnya, menggunakan nama layanan `com.amazonaws.Region.appmesh-envoy-management`) tidak didukung untuk koneksi Envoy.

Untuk pertimbangan dan batasan tambahan, lihat [Pertimbangan Zona Ketersediaan Titik Akhir Antarmuka dan Properti dan Batasan Titik Akhir Antarmuka](#).

Buat titik akhir VPC antarmuka untuk App Mesh

Untuk membuat titik akhir VPC antarmuka untuk layanan App Mesh, gunakan prosedur [Membuat Titik Akhir Antarmuka](#) di Panduan Pengguna Amazon VPC. Tentukan `com.amazonaws.Region.appmesh-envoy-management` nama layanan untuk proxy Envoy Anda untuk terhubung ke layanan manajemen Utusan publik App Mesh dan `com.amazonaws.Region.appmesh` untuk operasi mesh.

Note

Region mewakili pengenalan Wilayah untuk AWS Wilayah yang didukung oleh App Mesh, seperti `us-east-2` untuk Wilayah AS Timur (Ohio).

Meskipun Anda dapat menentukan titik akhir VPC antarmuka untuk App Mesh di Wilayah mana pun di mana App Mesh didukung, Anda mungkin tidak dapat menentukan titik akhir untuk semua Availability Zone di setiap Wilayah. Untuk mengetahui Availability Zones mana yang didukung dengan titik akhir VPC antarmuka di Wilayah, gunakan [describe-vpc-endpoint-services](#) perintah atau gunakan Konsol Manajemen AWS. Misalnya, perintah berikut mengembalikan zona ketersediaan tempat Anda dapat menerapkan titik akhir VPC antarmuka App Mesh di Wilayah AS Timur (Ohio):

```
aws --region us-east-2 ec2 describe-vpc-endpoint-services --query 'ServiceDetails[? ServiceName==`com.amazonaws.us-east-2.appmesh-envoy-management`].AvailabilityZones[]'
```

```
aws --region us-east-2 ec2 describe-vpc-endpoint-services --query 'ServiceDetails[? ServiceName==`com.amazonaws.us-east-2.appmesh`].AvailabilityZones[]'
```

Ketahanan di AWS App Mesh

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk AWS App Mesh. Setelah 30 September 2026, Anda tidak akan lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi

lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

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 serta 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.

App Mesh menjalankan instance bidang kontrolnya di beberapa Availability Zone untuk memastikan ketersediaan yang tinggi. App Mesh secara otomatis mendeteksi dan mengganti instance bidang kontrol yang tidak sehat, dan menyediakan peningkatan dan penambalan versi otomatis untuk mereka.

Pemulihan bencana di AWS App Mesh

Layanan App Mesh mengelola backup data pelanggan. Tidak ada yang perlu Anda lakukan untuk mengelola backup. Data yang dicadangkan dienkripsi.

Konfigurasi dan analisis kerentanan di AWS App Mesh

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk . AWS App Mesh Setelah 30 September 2026, Anda tidak akan lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

App Mesh menjual [image container Docker proxy Envoy](#) terkelola yang Anda terapkan dengan layanan mikro Anda. App Mesh memastikan bahwa gambar kontainer ditambah dengan patch kerentanan dan kinerja terbaru. App Mesh menguji rilis proxy Envoy baru terhadap set fitur App Mesh sebelum membuat gambar tersedia untuk Anda.

Anda harus memperbarui layanan mikro Anda untuk menggunakan versi gambar kontainer yang diperbarui. Berikut ini adalah versi terbaru dari gambar.

```
840364872350.dkr.ecr.region-code.amazonaws.com/aws-appmesh-envoy:v1.34.13.0-prod
```

Pemecahan masalah App Mesh

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk. AWS App Mesh Setelah 30 September 2026, Anda tidak lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Bab ini membahas pemecahan masalah praktik terbaik dan masalah umum yang mungkin Anda temui saat menggunakan App Mesh. Pilih salah satu area berikut untuk meninjau praktik terbaik dan masalah umum untuk area tersebut.

Topik

- [Praktik terbaik pemecahan masalah App Mesh](#)
- [Pemecahan masalah pengaturan App Mesh](#)
- [Pemecahan masalah konektivitas App Mesh](#)
- [Pemecahan masalah penskalaan App Mesh](#)
- [Pemecahan masalah observabilitas App Mesh](#)
- [Pemecahan masalah keamanan App Mesh](#)
- [Pemecahan masalah App Mesh Kubernetes](#)

Praktik terbaik pemecahan masalah App Mesh

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk. AWS App Mesh Setelah 30 September 2026, Anda tidak lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Kami menyarankan Anda mengikuti praktik terbaik dalam topik ini untuk memecahkan masalah saat menggunakan App Mesh.

Aktifkan antarmuka administrasi proxy Envoy

Proxy Envoy dikirimkan dengan antarmuka administrasi yang dapat Anda gunakan untuk menemukan konfigurasi dan statistik dan untuk melakukan fungsi administratif lainnya seperti pengeringan koneksi. Untuk informasi selengkapnya, lihat [Antarmuka administrasi](#) dalam dokumentasi Utusan.

Jika Anda menggunakan managed [Gambar utusan](#), endpoint administrasi diaktifkan secara default pada port 9901. Contoh yang diberikan dalam [Pemecahan masalah pengaturan App Mesh](#) menampilkan URL endpoint administrasi contoh sebagai `http://my-app.default.svc.cluster.local:9901/`.

Note

Titik akhir administrasi tidak boleh diekspos ke internet publik. Selain itu, kami merekomendasikan pemantauan log endpoint administrasi, yang ditetapkan oleh variabel `ENVOY_ADMIN_ACCESS_LOG_FILE` lingkungan secara `/tmp/envoy_admin_access.log` default.

Aktifkan integrasi Envoy DogStats D untuk pembongkaran metrik

Proxy Envoy dapat dikonfigurasi untuk membongkar statistik untuk lalu lintas OSI Layer 4 dan Layer 7 dan untuk kesehatan proses internal. Meskipun topik ini menunjukkan cara menggunakan statistik ini tanpa membongkar metrik ke sink seperti CloudWatch metrik dan Prometheus., memiliki statistik ini di lokasi terpusat untuk semua aplikasi Anda dapat membantu Anda mendiagnosis masalah dan mengonfirmasi perilaku lebih cepat. Untuk informasi selengkapnya, lihat [Menggunakan CloudWatch Metrik Amazon dan dokumentasi Prometheus](#).

Anda dapat mengonfigurasi metrik DogStats D dengan mengatur parameter yang ditentukan.

[DogStatsD variabel](#) Untuk informasi lebih lanjut tentang DogStats D, lihat dokumentasi [DogStatsD](#).

Anda dapat menemukan demonstrasi pembongkaran metrik ke AWS CloudWatch metrik di [App Mesh dengan panduan dasar-dasar Amazon ECS](#). [GitHub](#)

Aktifkan log akses

Sebaiknya aktifkan log akses pada Anda [Node virtual](#) dan [Gerbang virtual](#) untuk menemukan detail tentang lalu lintas transit di antara aplikasi Anda. Untuk informasi selengkapnya, lihat [Logging akses](#) di dokumentasi Utusan. Log memberikan informasi rinci tentang perilaku lalu lintas OSI Layer 4 dan Layer 7. Saat menggunakan format default Envoy, Anda dapat menganalisis log akses dengan [Wawasan CloudWatch Log](#) menggunakan pernyataan parse berikut.

```
parse @message "[*] \"* * *\" * * * * * * * * * * *\" as StartTime, Method, Path, Protocol, ResponseCode, ResponseFlags, BytesReceived, BytesSent, DurationMillis, UpstreamServiceTimeMillis, ForwardedFor, UserAgent, RequestId, Authority, UpstreamHost
```

Aktifkan logging debug Envoy di lingkungan pra-produksi

Sebaiknya atur level log proxy Envoy ke debug dalam lingkungan pra-produksi. Log debug dapat membantu Anda mengidentifikasi masalah sebelum Anda lulus konfigurasi App Mesh terkait ke lingkungan produksi Anda.

Jika Anda menggunakan [gambar Envoy](#), Anda dapat mengatur level log ke debug melalui variabel `ENVOY_LOG_LEVEL` lingkungan.

Note

Kami tidak merekomendasikan penggunaan debug level di lingkungan produksi. Menyetel level untuk debug meningkatkan pencatatan dan dapat memengaruhi kinerja dan biaya keseluruhan log yang diturunkan ke solusi seperti [CloudWatch Log](#).

Saat menggunakan format default Envoy, Anda dapat menganalisis log proses dengan [Wawasan CloudWatch Log](#) menggunakan pernyataan parse berikut:

```
parse @message "[*][*][*][*] [*] *" as Time, Thread, Level, Name, Source, Message
```

Pantau Konektivitas Proxy Utusan dengan bidang kontrol App Mesh

Sebaiknya Anda memantau metrik Envoy `control_plane.connected_state` untuk memastikan bahwa proxy Envoy berkomunikasi dengan bidang kontrol App Mesh untuk mengambil sumber daya konfigurasi dinamis. Untuk informasi selengkapnya, lihat [Server Manajemen](#).

Pemecahan masalah pengaturan App Mesh

⚠ Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk AWS App Mesh. Setelah 30 September 2026, Anda tidak lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Topik ini merinci masalah umum yang mungkin Anda alami dengan penyiapan App Mesh.

Tidak dapat menarik gambar wadah Utusan

Gejala

Anda menerima pesan galat berikut dalam tugas Amazon ECS. ECR Amazon *account ID* dan *Region* dalam pesan berikut mungkin berbeda, tergantung pada repositori Amazon ECR tempat Anda menarik gambar kontainer.

```
CannotPullContainerError: Error response from daemon: pull access denied for 840364872350.dkr.ecr.us-west-2.amazonaws.com/aws-appmesh-envoy, repository does not exist or may require 'docker login'
```

Resolusi

Kesalahan ini menunjukkan bahwa peran eksekusi tugas yang digunakan tidak memiliki izin untuk berkomunikasi ke Amazon ECR dan tidak dapat menarik gambar wadah Envoy dari repositori. Peran eksekusi tugas yang ditetapkan ke tugas Amazon ECS Anda memerlukan kebijakan IAM dengan pernyataan berikut:

```
{
  "Action": [
    "ecr:BatchCheckLayerAvailability",
    "ecr:GetDownloadUrlForLayer",
    "ecr:BatchGetImage"
  ],
  "Resource": "arn:aws:ecr:us-west-2:111122223333:repository/aws-appmesh-envoy",
  "Effect": "Allow"
}
```

```
},  
{  
  "Action": "ecr:GetAuthorizationToken",  
  "Resource": "*",  
  "Effect": "Allow"  
}
```

Jika masalah Anda masih belum teratasi, pertimbangkan untuk membuka [GitHub masalah](#) atau hubungi [AWS Support](#).

Tidak dapat terhubung ke layanan manajemen App Mesh Envoy

Gejala

Proxy Utusan Anda tidak dapat terhubung ke layanan manajemen Utusan App Mesh. Anda melihat:

- Koneksi ditolak kesalahan
- Batas waktu koneksi
- Kesalahan saat menyelesaikan titik akhir layanan manajemen Utusan App Mesh
- kesalahan gRPC

Resolusi

Pastikan proxy Utusan Anda memiliki akses ke internet atau ke [titik akhir VPC](#) pribadi dan [grup keamanan](#) Anda mengizinkan lalu lintas keluar di port 443. Titik akhir layanan manajemen Utusan publik App Mesh mengikuti format nama domain (FQDN) yang sepenuhnya memenuhi syarat.

```
# App Mesh Production Endpoint  
appmesh-envoy-management.Region-code.amazonaws.com  
  
# App Mesh Preview Endpoint  
appmesh-preview-envoy-management.Region-code.amazonaws.com
```

Anda dapat men-debug koneksi Anda ke EMS menggunakan perintah di bawah ini. Ini mengirimkan permintaan gRPC yang valid namun kosong ke Layanan Manajemen Utusan.

```
curl -v -k -H 'Content-Type: application/grpc' -X POST https://  
appmesh-envoy-management.Region-code.amazonaws.com:443/  
envoy.service.discovery.v3.AggregatedDiscoveryService/StreamAggregatedResources
```

Jika Anda menerima pesan ini kembali, koneksi Anda ke Layanan Manajemen Utusan berfungsi. Untuk men-debug kesalahan terkait gRPC, lihat kesalahan [di Utusan yang terputus dari layanan manajemen App Mesh Envoy](#) dengan teks kesalahan.

```
grpc-status: 16
grpc-message: Missing Authentication Token
```

Jika masalah Anda masih belum teratasi, pertimbangkan untuk membuka [GitHub masalah](#) atau hubungi [AWS Support](#).

Utusan terputus dari layanan manajemen App Mesh Envoy dengan teks kesalahan

Gejala

Proxy Envoy Anda tidak dapat terhubung ke layanan manajemen App Mesh Envoy dan menerima konfigurasinya. Log proxy Envoy Anda berisi entri log seperti berikut ini.

```
gRPC config stream closed: gRPC status code, message
```

Resolusi

Dalam kebanyakan kasus, bagian pesan dari log harus menunjukkan masalah. Tabel berikut mencantumkan kode status gRPC paling umum yang mungkin Anda lihat, penyebabnya, dan resolusinya.

kode status gRPC	Penyebab	Resolusi
0	Pemutusan hubungan yang anggun dari layanan manajemen Utusan.	Tidak ada masalah. App Mesh terkadang memutus proxy Envoy dengan kode status ini. Utusan akan terhubung kembali dan terus menerima pembaruan.
3	Endpoint mesh (virtual node atau virtual gateway), atau	Periksa kembali konfigurasi Envoy Anda untuk memastikan bahwa ia memiliki nama

kode status gRPC	Penyebab	Resolusi
	salah satu sumber daya yang terkait, tidak dapat ditemukan.	yang sesuai dari sumber daya App Mesh yang diwakilinya. Jika resource App Mesh Anda terintegrasi dengan AWS resource lain, seperti AWS Cloud Map ruang nama atau sertifikat ACM, pastikan sumber daya tersebut ada.
7	Proxy utusan tidak berwenang untuk melakukan tindakan, seperti terhubung ke layanan manajemen utusan, atau mengambil sumber daya terkait.	Pastikan Anda membuat kebijakan IAM yang memiliki pernyataan kebijakan yang sesuai untuk App Mesh dan layanan lainnya dan lampirkan kebijakan tersebut ke pengguna IAM atau peran yang digunakan proxy Utusan Anda untuk terhubung ke layanan manajemen Utusan.
8	Jumlah proxy Utusan untuk resource App Mesh tertentu melebihi kuota layanan tingkat akun.	Lihat Kuota layanan App Mesh informasi tentang kuota akun default dan cara meminta kenaikan kuota.

kode status gRPC	Penyebab	Resolusi
16	Proxy Envoy tidak memiliki kredensial otentikasi yang valid untuk AWS	Pastikan bahwa Utusan memiliki kredensial yang sesuai untuk terhubung ke AWS layanan melalui pengguna atau peran IAM. Masalah yang diketahui, #24136 , di Utusan untuk versi v1.24 dan sebelumnya gagal mengambil kredensial jika proses Envoy menggunakan lebih dari deskriptor file. 1024 Ini terjadi ketika Utusan melayani volume lalu lintas tinggi. Anda dapat mengonfirmasi masalah ini dengan memeriksa log Utusan di tingkat debug untuk teks <code>""</code> . A <code>libcurl</code> function was given a bad argument Untuk mengurangi masalah ini, tingkatkan ke versi Envoy atau yang lebih baru. <code>v1.25.1.0-prod</code>

Anda dapat mengamati kode status dan pesan dari proxy Utusan Anda dengan [Amazon CloudWatch Insights](#) menggunakan kueri berikut:

```
filter @message like /gRPC config stream closed/
| parse @message "gRPC config stream closed: *, *" as StatusCode, Message
```

Jika pesan kesalahan yang diberikan tidak membantu, atau masalah Anda masih belum teratasi, pertimbangkan untuk membuka [GitHub masalah](#).

Pemeriksaan kesehatan wadah utusan, pemeriksaan kesiapan, atau penyelidikan keaktifan gagal

Gejala

Proxy Envoy Anda gagal dalam pemeriksaan kesehatan dalam tugas Amazon ECS, instans Amazon EC2, atau pod Kubernetes. Misalnya, Anda menanyakan antarmuka administrasi Utusan dengan perintah berikut dan menerima status selain. LIVE

```
curl -s http://my-app.default.svc.cluster.local:9901/server_info | jq '.state'
```

Resolusi

Berikut ini adalah daftar langkah-langkah remediasi tergantung pada status yang dikembalikan oleh proxy Utusan.

- **PRE_INITIALIZING** atau **INITIALIZING** — Proxy Envoy belum menerima konfigurasi, atau tidak dapat menghubungkan dan mengambil konfigurasi dari layanan manajemen App Mesh Envoy. Utusan mungkin menerima kesalahan dari layanan manajemen utusan ketika mencoba untuk terhubung. Untuk informasi lebih lanjut, lihat kesalahan di [Utusan terputus dari layanan manajemen App Mesh Envoy dengan teks kesalahan](#).
- **DRAINING**— Proxy utusan telah mulai menguras koneksi sebagai tanggapan / `drain_listeners` atas permintaan `/healthcheck/fail` atau pada antarmuka administrasi Utusan. Kami tidak menyarankan untuk menjalankan jalur ini pada antarmuka administrasi kecuali Anda akan menghentikan tugas Amazon ECS, instans Amazon EC2, atau pod Kubernetes.

Jika masalah Anda masih belum teratasi, pertimbangkan untuk membuka [GitHub masalah](#) atau hubungi [AWS Support](#).

Pemeriksaan kesehatan dari penyeimbang beban ke titik akhir mesh gagal

Gejala

Titik akhir mesh Anda dianggap sehat oleh pemeriksaan kesehatan kontainer atau probe kesiapan, tetapi pemeriksaan kesehatan dari penyeimbang beban ke titik akhir mesh gagal.

Resolusi

Untuk mengatasi masalah ini, selesaikan tugas-tugas berikut.

- Pastikan bahwa [grup keamanan](#) yang terkait dengan titik akhir mesh Anda menerima lalu lintas masuk pada port yang telah Anda konfigurasi untuk pemeriksaan kesehatan Anda.
- Pastikan bahwa pemeriksaan kesehatan berhasil secara konsisten ketika diminta secara manual; misalnya, dari [host benteng dalam VPC](#) Anda.
- Jika Anda mengonfigurasi pemeriksaan kesehatan untuk node virtual, maka kami sarankan untuk menerapkan titik akhir pemeriksaan kesehatan di aplikasi Anda; misalnya, /ping untuk HTTP. Ini memastikan bahwa proxy Envoy dan aplikasi Anda dapat dirutekan dari penyeimbang beban.
- Anda dapat menggunakan jenis penyeimbang beban elastis apa pun untuk simpul virtual, tergantung pada fitur yang Anda butuhkan. Untuk informasi selengkapnya, lihat fitur [Elastic Load Balancing](#).
- Jika Anda mengonfigurasi pemeriksaan kesehatan untuk [gateway virtual](#), maka sebaiknya gunakan [penyeimbang beban jaringan](#) dengan pemeriksaan kesehatan TCP atau TLS pada port pendengar gateway virtual. Ini memastikan bahwa pendengar gateway virtual di-bootstrap dan siap menerima koneksi.

Jika masalah Anda masih belum teratasi, pertimbangkan untuk membuka [GitHub masalah](#) atau hubungi [AWS Support](#).

Gateway virtual tidak menerima lalu lintas di port 1024 atau kurang

Gejala

Gateway virtual Anda tidak menerima lalu lintas di port 1024 atau kurang, tetapi menerima lalu lintas pada nomor port yang lebih besar dari 1024. Misalnya, Anda menanyakan statistik Utusan dengan perintah berikut dan menerima nilai selain nol.

```
curl -s http://my-app.default.svc.cluster.local:9901/stats | grep "update_rejected"
```

Anda mungkin melihat teks yang mirip dengan teks berikut di log Anda yang menjelaskan kegagalan untuk mengikat ke port istimewa:

```
gRPC config for type.googleapis.com/envoy.api.v2.Listener rejected: Error adding/
updating listener(s) lds_ingress_0.0.0.0_port_<port num>: cannot bind '0.0.0.0:<port
num>': Permission denied
```

Resolusi

Untuk mengatasi masalah ini, pengguna yang ditentukan untuk gateway harus memiliki kemampuan linuxCAP_NET_BIND_SERVICE. Untuk informasi selengkapnya, lihat [Capabilities](#) in the Linux Programmer's Manual, [parameter Linux dalam parameter](#) definisi Tugas ECS, dan [Menetapkan kapabilitas untuk sebuah container](#) dalam dokumentasi Kubernetes.

⚠ Important

Fargate harus menggunakan nilai port yang lebih besar dari 1024.

Jika masalah Anda masih belum teratasi, pertimbangkan untuk membuka [GitHub masalah](#) atau hubungi [AWS Support](#).

Pemecahan masalah konektivitas App Mesh

⚠ Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk AWS App Mesh. Setelah 30 September 2026, Anda tidak lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Topik ini merinci masalah umum yang mungkin Anda alami dengan konektivitas App Mesh.

Tidak dapat menyelesaikan nama DNS untuk layanan virtual

Gejala

Aplikasi Anda tidak dapat menyelesaikan nama DNS dari layanan virtual yang sedang dicoba untuk terhubung.

Resolusi

Ini adalah masalah yang diketahui. Untuk informasi selengkapnya, lihat [Nama VirtualServices berdasarkan nama host atau masalah GitHub FQDN apa pun](#). Layanan virtual di App Mesh dapat diberi nama apa saja. Selama ada A catatan DNS untuk nama layanan virtual dan aplikasi dapat menyelesaikan nama layanan virtual, permintaan akan diproksi oleh Utusan dan diarahkan ke tujuan

yang sesuai. Untuk mengatasi masalah ini, tambahkan A catatan DNS ke alamat IP non-loopback, seperti `10.10.10.10`, untuk nama layanan virtual. A Catatan DNS dapat ditambahkan dalam kondisi berikut:

- Di Amazon Route 53, jika nama diakhiran dengan nama zona host pribadi Anda
- Di dalam `/etc/hosts` file wadah aplikasi
- Di server DNS pihak ketiga yang Anda kelola

Jika masalah Anda masih belum teratasi, pertimbangkan untuk membuka [GitHub masalah](#) atau hubungi [AWS Support](#).

Tidak dapat terhubung ke backend layanan virtual

Gejala

Aplikasi Anda tidak dapat membuat koneksi ke layanan virtual yang didefinisikan sebagai backend pada node virtual Anda. Saat mencoba membuat koneksi, koneksi mungkin gagal sepenuhnya, atau permintaan dari perspektif aplikasi mungkin gagal dengan kode HTTP `503` respons.

Resolusi

Jika aplikasi gagal terhubung sama sekali (tidak ada kode HTTP `503` respons yang dikembalikan), maka lakukan hal berikut:

- Pastikan bahwa lingkungan komputasi Anda telah diatur agar berfungsi dengan App Mesh.
 - Untuk Amazon ECS, pastikan Anda mengaktifkan [konfigurasi proxy](#) yang sesuai. Untuk end-to-end penelusuran, lihat [Memulai dengan App Mesh dan Amazon ECS](#).
 - Untuk Kubernetes, termasuk Amazon EKS, pastikan Anda memiliki pengontrol App Mesh terbaru yang diinstal melalui Helm. Untuk informasi selengkapnya, lihat [App Mesh Controller](#) di Helm Hub atau [Tutorial: Konfigurasi integrasi App Mesh dengan Kubernetes](#).
 - Untuk Amazon EC2, pastikan Anda telah menyiapkan instans Amazon EC2 untuk memproksi lalu lintas App Mesh. Untuk informasi selengkapnya, lihat [Memperbarui layanan](#).
- Pastikan bahwa container Envoy yang berjalan pada layanan komputasi Anda telah berhasil terhubung ke layanan manajemen App Mesh Envoy. Anda dapat mengonfirmasi ini dengan memeriksa statistik Utusan untuk bidang tersebut. `control_plane.connected_state` Untuk informasi selengkapnya `control_plane.connected_state`, lihat [Memantau Konektivitas Proksi Utusan](#) di Praktik Terbaik Pemecahan Masalah kami.

Jika Utusan dapat membuat koneksi pada awalnya, tetapi kemudian terputus dan tidak pernah terhubung kembali, lihat Utusan [terputus dari layanan manajemen App Mesh Envoy dengan teks kesalahan untuk memecahkan masalah mengapa itu terputus](#).

Jika aplikasi terhubung tetapi permintaan gagal dengan kode HTTP 503 respons, coba yang berikut ini:

- Pastikan bahwa layanan virtual yang Anda sambungkan ada di mesh.
- Pastikan bahwa layanan virtual memiliki penyedia (router virtual atau node virtual).
- Saat menggunakan Envoy sebagai Proxy HTTP, jika Anda melihat lalu lintas keluar masuk `cluster.cds_egress_*_mesh-allow-all` alih-alih tujuan yang benar melalui statistik Envoy, kemungkinan besar Envoy tidak merutekan permintaan dengan benar. `filter_chains` Ini bisa disebabkan oleh penggunaan nama layanan virtual yang tidak memenuhi syarat. Kami menyarankan Anda menggunakan nama penemuan layanan dari layanan yang sebenarnya sebagai nama layanan virtual, karena proxy Envoy berkomunikasi dengan layanan virtual lainnya melalui nama mereka.

Untuk informasi selengkapnya, lihat [layanan virtual](#).

- Periksa log proxy Envoy untuk salah satu pesan galat berikut:
 - `No healthy upstream`— Node virtual yang coba dirutekan oleh proxy Envoy tidak memiliki titik akhir yang diselesaikan, atau tidak memiliki titik akhir yang sehat. Pastikan bahwa node virtual target memiliki penemuan layanan dan pengaturan pemeriksaan kesehatan yang benar.

Jika permintaan ke layanan gagal selama penerapan atau penskalaan layanan virtual backend, ikuti panduan di [Beberapa permintaan gagal dengan kode status HTTP 503 ketika layanan virtual memiliki penyedia node virtual](#)

- `No cluster match for URL` Ini kemungkinan besar disebabkan ketika permintaan dikirim ke layanan virtual yang tidak sesuai dengan kriteria yang ditentukan oleh salah satu rute yang ditentukan di bawah penyedia router virtual. Pastikan bahwa permintaan dari aplikasi dikirim ke rute yang didukung dengan memastikan jalur dan header permintaan HTTP sudah benar.
- `No matching filter chain found`— Ini kemungkinan besar disebabkan ketika permintaan dikirim ke layanan virtual pada port yang tidak valid. Pastikan bahwa permintaan dari aplikasi menggunakan port yang sama yang ditentukan pada router virtual.

Jika masalah Anda masih belum teratasi, pertimbangkan untuk membuka [GitHub masalah](#) atau hubungi [AWS Support](#).

Tidak dapat terhubung ke layanan eksternal

Gejala

Aplikasi Anda tidak dapat terhubung ke layanan di luar mesh, seperti `amazon.com`.

Resolusi

Secara default, App Mesh tidak mengizinkan lalu lintas keluar dari aplikasi di dalam mesh ke tujuan mana pun di luar mesh. Untuk mengaktifkan komunikasi dengan layanan eksternal, ada dua opsi:

- Atur [filter keluar](#) pada sumber daya mesh ke `ALLOW_ALL`. Pengaturan ini akan memungkinkan aplikasi apa pun di dalam mesh untuk berkomunikasi dengan alamat IP tujuan apa pun di dalam atau di luar mesh.
- Model layanan eksternal di mesh menggunakan layanan virtual, router virtual, rute, dan node virtual. Misalnya, untuk memodelkan layanan eksternal `example.com`, Anda dapat membuat layanan virtual bernama `example.com` dengan router virtual dan rute yang mengirimkan semua lalu lintas ke node virtual dengan nama host penemuan layanan DNS dari `example.com`

Jika masalah Anda masih belum teratasi, pertimbangkan untuk membuka [GitHub masalah](#) atau hubungi [AWS Support](#).

Tidak dapat terhubung ke server MySQL atau SMTP

Gejala

Saat mengizinkan lalu lintas keluar ke semua tujuan (`Mesh EgressFilter type =ALLOW_ALL`), seperti server SMTP atau database MySQL menggunakan definisi node virtual, koneksi dari aplikasi Anda gagal. Sebagai contoh, berikut ini adalah pesan kesalahan dari mencoba untuk terhubung ke server MySQL.

```
ERROR 2013 (HY000): Lost connection to MySQL server at 'reading initial communication packet', system error: 0
```

Resolusi

Ini adalah masalah yang diketahui yang diselesaikan dengan menggunakan gambar App Mesh versi 1.15.0 atau yang lebih baru. Untuk informasi selengkapnya, lihat masalah [Tidak dapat terhubung ke MySQL dengan App Mesh](#). GitHub Kesalahan ini terjadi karena listener keluar di Envoy yang dikonfigurasi oleh App Mesh menambahkan filter listener Envoy TLS Inspector. Untuk informasi selengkapnya, lihat [TLS Inspector](#) di dokumentasi Envoy. Filter ini mengevaluasi apakah koneksi menggunakan TLS atau tidak dengan memeriksa paket pertama yang dikirim dari klien. Dengan MySQL dan SMTP, bagaimanapun, server mengirimkan paket pertama setelah koneksi. Untuk informasi selengkapnya tentang MySQL, [lihat Jabat Tangan Awal](#) dalam dokumentasi MySQL. Karena server mengirimkan paket pertama, inspeksi pada filter gagal.

Untuk mengatasi masalah ini tergantung pada versi Utusan Anda:

- Jika versi Envoy image App Mesh Anda adalah 1.15.0 atau yang lebih baru, jangan memodelkan layanan eksternal seperti MySQL, SMTP, MSSQL, dll. sebagai backend untuk node virtual aplikasi Anda.
- Jika versi Envoy image App Mesh Anda sebelum 1.15.0, tambahkan port **3306** ke daftar nilai untuk layanan Anda untuk MySQL dan sebagai port yang Anda gunakan untuk STMP.
APPESH_EGRESS_IGNORED_PORTS

Important

Sementara port SMTP standar adalah 25, dan 587465, Anda hanya harus menambahkan port yang Anda gunakan APPESH_EGRESS_IGNORED_PORTS dan tidak ketiganya.

Untuk informasi selengkapnya, lihat [Update services](#) for Kubernetes, [Update services for](#) Amazon ECS, atau [Update services for Amazon](#) EC2.

Jika masalah Anda masih belum teratasi, Anda dapat memberi kami detail tentang apa yang Anda alami menggunakan [GitHub masalah](#) yang ada atau hubungi [AWS Support](#).

Tidak dapat terhubung ke layanan yang dimodelkan sebagai node virtual TCP atau router virtual di App Mesh

Gejala

Aplikasi Anda tidak dapat terhubung ke backend yang menggunakan pengaturan protokol TCP dalam definisi App Mesh. [PortMapping](#)

Resolusi

Ini adalah masalah yang diketahui. Untuk informasi selengkapnya, lihat [Perutean ke beberapa tujuan TCP pada port yang sama](#). GitHub App Mesh saat ini tidak mengizinkan beberapa tujuan backend yang dimodelkan sebagai TCP untuk berbagi port yang sama karena pembatasan informasi yang diberikan ke proxy Utusan di Lapisan OSI 4. Untuk memastikan bahwa lalu lintas TCP dapat dirutekan dengan tepat untuk semua tujuan backend, lakukan hal berikut:

- Pastikan bahwa semua tujuan menggunakan port yang unik. Jika Anda menggunakan penyedia router virtual untuk layanan virtual backend, Anda dapat mengubah port router virtual tanpa mengubah port pada node virtual yang dituju. Ini memungkinkan aplikasi untuk membuka koneksi pada port router virtual sementara proxy Envoy terus menggunakan port yang ditentukan dalam node virtual.
- Jika tujuan yang dimodelkan sebagai TCP adalah server MySQL, atau protokol berbasis TCP lainnya di mana server mengirim paket pertama setelah koneksi, lihat. [Tidak dapat terhubung ke server MySQL atau SMTP](#)

Jika masalah Anda masih belum teratasi, Anda dapat memberi kami detail tentang apa yang Anda alami menggunakan [GitHub masalah](#) yang ada atau hubungi [AWS Support](#).

Konektivitas berhasil ke layanan yang tidak terdaftar sebagai backend layanan virtual untuk node virtual

Gejala

Aplikasi Anda dapat menghubungkan dan mengirim lalu lintas ke tujuan yang tidak ditentukan sebagai backend layanan virtual pada node virtual Anda.

Resolusi

Jika permintaan berhasil ke tujuan yang belum dimodelkan di App Mesh APIs, maka penyebab yang paling mungkin adalah bahwa jenis [filter keluar](#) mesh telah disetel ke. ALLOW_ALL Ketika filter keluar diatur keALLOW_ALL, permintaan keluar dari aplikasi Anda yang tidak cocok dengan tujuan model (backend) akan dikirim ke alamat IP tujuan yang ditetapkan oleh aplikasi.

Jika Anda ingin melarang lalu lintas ke tujuan yang tidak dimodelkan di mesh, pertimbangkan untuk mengatur nilai filter keluar ke. DROP_ALL

Note

Mengatur nilai filter keluar mesh mempengaruhi semua node virtual di dalam mesh. Mengkonfigurasi `egress_filter` sebagai `DROP_ALL` dan mengaktifkan TLS tidak tersedia untuk lalu lintas keluar yang tidak ke domain. AWS

Jika masalah Anda masih belum teratasi, pertimbangkan untuk membuka [GitHub masalah](#) atau hubungi [AWS Support](#).

Beberapa permintaan gagal dengan kode status HTTP **503** ketika layanan virtual memiliki penyedia node virtual

Gejala

Sebagian dari permintaan aplikasi Anda gagal ke backend layanan virtual yang menggunakan penyedia node virtual, bukan penyedia router virtual. Saat menggunakan penyedia router virtual untuk layanan virtual, permintaan tidak gagal.

Resolusi

Ini adalah masalah yang diketahui. Untuk informasi selengkapnya, lihat [Kebijakan coba lagi tentang penyedia Node Virtual untuk Layanan Virtual](#) di GitHub. Saat menggunakan node virtual sebagai penyedia layanan virtual, Anda tidak dapat menentukan kebijakan coba ulang default yang Anda inginkan untuk digunakan oleh klien layanan virtual Anda. Sebagai perbandingan, penyedia router virtual memungkinkan kebijakan coba ulang ditentukan karena mereka adalah properti dari sumber daya rute anak.

Untuk mengurangi kegagalan permintaan ke penyedia node virtual, gunakan penyedia router virtual sebagai gantinya, dan tentukan kebijakan coba lagi pada rutenya. Untuk cara lain untuk mengurangi kegagalan permintaan pada aplikasi Anda, lihat [Praktik terbaik App Mesh](#).

Jika masalah Anda masih belum teratasi, pertimbangkan untuk membuka [GitHub masalah](#) atau hubungi [AWS Support](#).

Tidak dapat terhubung ke sistem file Amazon EFS

Gejala

Saat mengonfigurasi tugas Amazon ECS dengan sistem file Amazon EFS sebagai volume, tugas gagal dimulai dengan kesalahan berikut.

```
ResourceInitializationError: failed to invoke EFS utils commands to set up EFS volumes:  
stderr: mount.nfs4: Connection refused : unsuccessful EFS utils command execution;  
code: 32
```

Resolusi

Ini adalah masalah yang diketahui. Kesalahan ini terjadi karena koneksi NFS ke Amazon EFS terjadi sebelum kontainer apa pun dalam tugas Anda dimulai. Lalu lintas ini dirutekan oleh konfigurasi proxy ke Envoy, yang tidak akan berjalan pada saat ini. Karena pemesanan startup, klien NFS gagal terhubung ke sistem file Amazon EFS dan tugas gagal diluncurkan. Untuk mengatasi masalah ini, tambahkan port 2049 ke daftar nilai untuk `EgressIgnoredPorts` pengaturan dalam konfigurasi proxy definisi tugas Amazon ECS Anda. Untuk informasi selengkapnya, lihat [Konfigurasi proxy](#).

Jika masalah Anda masih belum teratasi, pertimbangkan untuk membuka [GitHub masalah](#) atau hubungi [AWS Support](#).

Konektivitas berhasil dilayani, tetapi permintaan yang masuk tidak muncul di log akses, jejak, atau metrik untuk Utusan

Gejala

Meskipun aplikasi Anda dapat menghubungkan dan mengirim permintaan ke aplikasi lain, Anda tidak dapat melihat permintaan masuk di log akses atau dalam melacak informasi untuk proxy Utusan.

Resolusi

Ini adalah masalah yang diketahui. Dari informasi selengkapnya, lihat masalah [penyiapan aturan iptables](#) di Github. Proxy Envoy hanya mencegat lalu lintas masuk ke port tempat node virtual yang sesuai mendengarkan. Permintaan ke port lain akan melewati proxy Envoy dan mencapai layanan di belakangnya secara langsung. Untuk membiarkan proxy Envoy mencegat lalu lintas masuk untuk layanan Anda, Anda perlu mengatur node virtual dan layanan Anda untuk mendengarkan pada port yang sama.

Jika masalah Anda masih belum teratasi, pertimbangkan untuk membuka [GitHub masalah](#) atau hubungi [AWS Support](#).

Menyetel variabel `HTTP_PROXY/HTTPS_PROXY` environment pada level container tidak berfungsi seperti yang diharapkan.

Gejala

Ketika `HTTP_PROXY/HTTPS_PROXY` disetel sebagai variabel lingkungan di:

- Penampung aplikasi dalam definisi tugas dengan App Mesh diaktifkan, permintaan yang dikirim ke namespace layanan App Mesh akan mendapatkan respons HTTP 500 kesalahan dari sespan Envoy.
- Wadah utusan dalam definisi tugas dengan App Mesh diaktifkan, permintaan yang keluar dari sespan Envoy tidak akan melalui server HTTPS proxy HTTP/, dan variabel lingkungan tidak akan berfungsi.

Resolusi

Untuk wadah aplikasi:

App Mesh berfungsi dengan memiliki lalu lintas dalam tugas Anda melalui proxy Envoy. `HTTP_PROXY/HTTPS_PROXY` konfigurasi mengesampingkan perilaku ini dengan mengonfigurasi lalu lintas kontainer untuk melewati proxy eksternal yang berbeda. Lalu lintas masih akan dicegat oleh Utusan, tetapi tidak mendukung proxy lalu lintas mesh menggunakan proxy eksternal.

Jika Anda ingin mem-proxy semua lalu lintas non-mesh, setel `NO_PROXY` untuk menyertakan CIDR/ Namespace mesh Anda, localhost, dan titik akhir kredensial seperti pada contoh berikut.

```
NO_PROXY=localhost,127.0.0.1,169.254.169.254,169.254.170.2,10.0.0.0/16
```

Untuk wadah Utusan:

Utusan tidak mendukung proxy generik. Kami tidak menyarankan pengaturan variabel-variabel ini.

Jika masalah Anda masih belum teratasi, pertimbangkan untuk membuka [GitHub masalah](#) atau hubungi [AWS Support](#).

Batas waktu permintaan hulu bahkan setelah mengatur batas waktu untuk `route`.

Gejala

Anda menentukan batas waktu untuk:

- Rute, tetapi Anda masih mendapatkan kesalahan batas waktu permintaan hulu.
- Pendengar simpul virtual dan batas waktu serta batas waktu coba lagi untuk rute, tetapi Anda masih mendapatkan kesalahan batas waktu permintaan hulu.

Resolusi

Agar permintaan latensi tinggi lebih dari 15 detik berhasil diselesaikan, Anda perlu menentukan batas waktu di tingkat pendengar rute dan node virtual.

Jika Anda menentukan batas waktu rute yang lebih besar dari default 15 detik, pastikan batas waktu juga ditentukan untuk pendengar untuk semua node virtual yang berpartisipasi. Namun, jika Anda mengurangi batas waktu ke nilai yang lebih rendah dari default, itu opsional untuk memperbarui batas waktu di node virtual. Untuk informasi selengkapnya tentang opsi saat menyiapkan node dan rute [virtual](#), lihat [node](#) dan [rute](#) virtual.

Jika Anda menetapkan kebijakan coba lagi, durasi yang Anda tentukan untuk batas waktu permintaan harus selalu lebih besar dari atau sama dengan batas waktu coba lagi dikalikan dengan percobaan ulang maksimal yang Anda tentukan dalam kebijakan coba lagi. Ini memungkinkan permintaan Anda dengan semua percobaan ulang berhasil diselesaikan. Untuk informasi selengkapnya, lihat [rute](#).

Jika masalah Anda masih belum teratasi, pertimbangkan untuk membuka [GitHub masalah](#) atau hubungi [AWS Support](#).

Utusan merespons dengan permintaan HTTP Bad.

Gejala

Utusan merespons dengan HTTP 400 Bad request untuk semua permintaan yang dikirim melalui Network Load Balancer (NLB). Saat kami memeriksa log Utusan, kami melihat:

- Log debug:

```
dispatch error: http/1.1 protocol error: HPE_INVALID_METHOD
```

- Akses log:

```
"- - HTTP/1.1" 400 DPE 0 11 0 - "-" "-" "-" "-" "
```

Resolusi

Resolusinya adalah menonaktifkan protokol proxy versi 2 (PPv2) pada atribut [grup target](#) NLB Anda.

Sampai hari ini tidak PPv2 didukung oleh gateway virtual dan utusan node virtual yang dijalankan menggunakan bidang kontrol App Mesh. Jika Anda menerapkan NLB menggunakan pengontrol penyeimbang AWS beban di Kubernetes, maka nonaktifkan PPv2 dengan menyetel atribut berikut ke: `false`

```
service.beta.kubernetes.io/aws-load-balancer-target-group-attributes:  
  proxy_protocol_v2.enabled
```

Lihat [Anotasi Pengontrol AWS Load Balancer untuk detail selengkapnya](#) tentang atribut sumber daya NLB.

Jika masalah Anda masih belum teratasi, pertimbangkan untuk membuka [GitHub masalah](#) atau hubungi [AWS Support](#).

Tidak dapat mengonfigurasi batas waktu dengan benar.

Gejala

Batas waktu permintaan Anda dalam waktu 15 detik bahkan setelah mengonfigurasi batas waktu pada pendengar node virtual dan batas waktu pada rute menuju backend node virtual.

Resolusi

Pastikan bahwa layanan virtual yang benar termasuk dalam daftar backend.

Jika masalah Anda masih belum teratasi, pertimbangkan untuk membuka [GitHub masalah](#) atau hubungi [AWS Support](#).

Pemecahan masalah penskalaan App Mesh

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk. AWS App Mesh Setelah 30 September 2026, Anda tidak lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi

lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Topik ini merinci masalah umum yang mungkin Anda alami dengan penskalaan App Mesh.

Konektivitas gagal dan pemeriksaan kesehatan kontainer gagal saat menskalakan melebihi 50 replika untuk gateway virtual node/virtual

Gejala

Saat Anda menskalakan jumlah replika, seperti tugas Amazon ECS, pod Kubernetes, atau instans Amazon EC2, untuk node/virtual gateway virtual di atas 50, pemeriksaan kesehatan container Envoy untuk Utusan baru dan yang sedang berjalan mulai gagal. Aplikasi hilir yang mengirimkan lalu lintas ke node/virtual gateway virtual mulai melihat kegagalan permintaan dengan kode 503 status HTTP.

Resolusi

Kuota default App Mesh untuk jumlah Utusan per node/virtual gateway virtual adalah 50. Ketika jumlah Utusan yang berjalan melebihi kuota ini, Utusan baru dan yang sedang berjalan gagal terhubung ke layanan manajemen Utusan App Mesh dengan kode status gRPC (`RESOURCE_EXHAUSTED`). Kuota ini bisa dinaikkan. Untuk informasi selengkapnya, lihat [Kuota layanan App Mesh](#).

Jika masalah Anda masih belum teratasi, pertimbangkan untuk membuka [GitHub masalah](#) atau hubungi [AWS Support](#).

Permintaan gagal **503** ketika backend layanan virtual secara horizontal keluar atau masuk

Gejala

Ketika layanan virtual backend secara horizontal diskalakan atau masuk, permintaan dari aplikasi hilir gagal dengan kode status HTTP `503`.

Resolusi

App Mesh merekomendasikan beberapa pendekatan untuk mengurangi kasus kegagalan saat menskalakan aplikasi secara horizontal. Untuk informasi terperinci tentang cara mencegah kegagalan ini, lihat [Praktik terbaik App Mesh](#).

Jika masalah Anda masih belum teratasi, pertimbangkan untuk membuka [GitHub masalah](#) atau hubungi [AWS Support](#).

Kontainer utusan mogok dengan segfault di bawah peningkatan beban

Gejala

Di bawah beban lalu lintas yang tinggi, proxy Envoy macet karena kesalahan segmentasi (kode keluar Linux). 139 Log proses Envoy berisi pernyataan seperti berikut ini.

```
Caught Segmentation fault, suspect faulting address 0x0"
```

Resolusi

Proxy Envoy kemungkinan telah melanggar standar nofile ulimit sistem operasi, batas jumlah file yang dapat dibuka suatu proses pada suatu waktu. Pelanggaran ini disebabkan oleh lalu lintas yang menyebabkan lebih banyak koneksi, yang mengkonsumsi soket sistem operasi tambahan. Untuk mengatasi masalah ini, tingkatkan nilai nofile ulimit pada sistem operasi host. Jika Anda menggunakan Amazon ECS, batas ini dapat diubah melalui pengaturan [Ulimit pada pengaturan batas sumber daya](#) definisi tugas.

Jika masalah Anda masih belum teratasi, pertimbangkan untuk membuka [GitHub masalah](#) atau hubungi [AWS Support](#).

Peningkatan sumber daya default tidak tercermin dalam Batas Layanan

Gejala

Setelah meningkatkan batas default resource App Mesh, nilai baru tidak tercermin saat Anda melihat batas layanan Anda.

Resolusi

Meskipun batas baru saat ini tidak ditampilkan, pelanggan masih dapat menggunakannya.

Jika masalah Anda masih belum teratasi, pertimbangkan untuk membuka [GitHub masalah](#) atau hubungi [AWS Support](#).

Aplikasi macet karena sejumlah besar panggilan pemeriksaan kesehatan.

Gejala

Setelah mengaktifkan pemeriksaan kesehatan aktif untuk node virtual, ada peningkatan jumlah panggilan pemeriksaan kesehatan. Aplikasi mogok karena volume panggilan pemeriksaan kesehatan yang sangat meningkat yang dilakukan ke aplikasi.

Resolusi

Ketika pemeriksaan kesehatan aktif diaktifkan, setiap titik akhir Utusan dari hilir (klien) mengirimkan permintaan kesehatan ke setiap titik akhir cluster hulu (server) untuk membuat keputusan perutean. Akibatnya jumlah total permintaan pemeriksaan kesehatan akan $\text{number of client Envoy} \times \text{active health check frequency}$.

Untuk mengatasi masalah ini, modifikasi frekuensi pemeriksaan kesehatan, yang akan mengurangi volume total pemeriksaan kesehatan. Selain pemeriksaan kesehatan aktif, App Mesh memungkinkan konfigurasi [deteksi outlier](#) sebagai sarana pemeriksaan kesehatan pasif. Gunakan deteksi outlier untuk mengonfigurasi kapan harus menghapus host tertentu berdasarkan respons yang berurutan 5xx.

Jika masalah Anda masih belum teratasi, pertimbangkan untuk membuka [GitHub masalah](#) atau hubungi [AWS Support](#).

Pemecahan masalah observabilitas App Mesh

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk AWS App Mesh. Setelah 30 September 2026, Anda tidak lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Topik ini merinci masalah umum yang mungkin Anda alami dengan observabilitas App Mesh.

Tidak dapat melihat AWS X-Ray jejak untuk aplikasi saya

Gejala

Aplikasi Anda di App Mesh tidak menampilkan informasi penelusuran X-Ray di konsol X-Ray atau APIs

Resolusi

Untuk menggunakan X-Ray di App Mesh, Anda harus mengonfigurasi komponen dengan benar untuk mengaktifkan komunikasi antara aplikasi, wadah sespan, dan layanan X-Ray. Ambil langkah-langkah berikut untuk mengonfirmasi bahwa X-Ray telah diatur dengan benar:

- Pastikan protokol pendengar Node Virtual App Mesh tidak disetel sebagai TCP.
- Pastikan bahwa wadah X-Ray yang digunakan dengan aplikasi Anda mengekspos port UDP 2000 dan berjalan sebagai pengguna. 1337 Untuk informasi selengkapnya, lihat [contoh X-Ray Amazon ECS](#) di GitHub.
- Pastikan bahwa wadah Envoy telah mengaktifkan tracing. Jika Anda menggunakan [image App Mesh Envoy](#), Anda dapat mengaktifkan X-Ray dengan menyetel variabel `ENABLE_ENVOY_XRAY_TRACING` lingkungan ke nilai 1 dan variabel `XRAY_DAEMON_PORT` lingkungan. 2000
- Jika Anda telah menginstrumentasi X-Ray dalam kode aplikasi Anda dengan salah satu [bahasa tertentu SDKs](#), maka pastikan bahwa itu dikonfigurasi dengan benar dengan mengikuti panduan untuk bahasa Anda.
- Jika semua item sebelumnya dikonfigurasi dengan benar, tinjau log kontainer X-Ray untuk kesalahan dan ikuti panduan dalam [Pemecahan Masalah AWS X-Ray](#). Penjelasan lebih rinci tentang integrasi X-Ray di App Mesh dapat ditemukan di [Mengintegrasikan X-Ray dengan App Mesh](#).

Jika masalah Anda masih belum teratasi, pertimbangkan untuk membuka [GitHub masalah](#) atau hubungi [AWS Support](#).

Tidak dapat melihat metrik Utusan untuk aplikasi saya di metrik Amazon CloudWatch

Gejala

Aplikasi Anda di App Mesh tidak memancarkan metrik yang dihasilkan oleh proxy Envoy ke metrik CloudWatch

Resolusi

Saat Anda menggunakan CloudWatch metrik di App Mesh, Anda harus mengonfigurasi beberapa komponen dengan benar untuk mengaktifkan komunikasi antara proxy Envoy, sespan CloudWatch

agen, dan layanan metrik. CloudWatch Ambil langkah-langkah berikut untuk mengonfirmasi bahwa CloudWatch metrik untuk proxy Envoy telah diatur dengan benar:

- Pastikan Anda menggunakan image CloudWatch agen untuk App Mesh. Untuk informasi selengkapnya, lihat [CloudWatchagen App Mesh](#) di GitHub.
- Pastikan Anda telah mengonfigurasi CloudWatch agen untuk App Mesh dengan tepat dengan mengikuti petunjuk penggunaan khusus platform. Untuk informasi selengkapnya, lihat [CloudWatchagen App Mesh](#) di GitHub.
- Jika semua item sebelumnya dikonfigurasi dengan benar, tinjau log kontainer CloudWatch agen untuk kesalahan dan ikuti panduan yang diberikan dalam [Pemecahan Masalah](#) agen. CloudWatch

Jika masalah Anda masih belum teratasi, pertimbangkan untuk membuka [GitHub masalah](#) atau hubungi [AWS Support](#).

Tidak dapat mengonfigurasi aturan pengambilan sampel khusus untuk jejak AWS X-Ray

Gejala

Aplikasi Anda menggunakan penelusuran X-Ray, tetapi Anda tidak dapat mengonfigurasi aturan pengambilan sampel untuk jejak Anda.

Resolusi

Karena App Mesh Envoy saat ini tidak mendukung konfigurasi pengambilan sampel X-Ray Dinamis, solusi berikut tersedia.

Jika versi Utusan Anda 1.19.1 atau lebih baru, Anda memiliki opsi berikut.

- Untuk hanya mengatur laju pengambilan sampel, gunakan variabel `XRAY_SAMPLING_RATE` lingkungan pada wadah Envoy. Nilai harus ditentukan sebagai desimal antara 0 dan 1.00 (100%). Untuk informasi selengkapnya, lihat [AWS X-Ray variabel](#).
- Untuk mengonfigurasi aturan pengambilan sampel kustom yang dilokalkan untuk pelacak X-Ray, gunakan variabel `XRAY_SAMPLING_RULE_MANIFEST` lingkungan untuk menentukan jalur file dalam sistem file kontainer Envoy. Untuk informasi selengkapnya, lihat [Aturan pengambilan sampel](#) di Panduan AWS X-Ray Pengembang.

Jika versi Utusan Anda sebelumnya 1.19.1, lakukan hal berikut.

- Gunakan variabel `ENVOY_TRACING_CFG_FILE` lingkungan untuk mengubah laju pengambilan sampel Anda. Untuk informasi selengkapnya, lihat [Variabel konfigurasi utusan](#). Tentukan konfigurasi penelusuran khusus dan tentukan aturan pengambilan sampel lokal. Untuk informasi selengkapnya, lihat konfigurasi [X-Ray Utusan](#).
- Konfigurasi penelusuran khusus untuk contoh variabel `ENVOY_TRACING_CFG_FILE` lingkungan:

```
tracing:
  http:
    name: envoy.tracers.xray
    typedConfig:
      "@type": type.googleapis.com/envoy.config.trace.v3.XRayConfig
      segmentName: foo/bar
      segmentFields:
        origin: AWS::AppMesh::Proxy
        aws:
          app_mesh:
            mesh_name: foo
            virtual_node_name: bar
      daemonEndpoint:
        protocol: UDP
        address: 127.0.0.1
        portValue: 2000
      samplingRuleManifest:
        filename: /tmp/sampling-rules.json
```

- Untuk detail tentang konfigurasi manifes aturan pengambilan sampel di `samplingRuleManifest` properti, lihat [Mengonfigurasi X-Ray SDK](#) for Go.

Jika masalah Anda masih belum teratasi, pertimbangkan untuk membuka [GitHub masalah](#) atau hubungi [AWS Support](#).

Pemecahan masalah keamanan App Mesh

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk. AWS App Mesh Setelah 30 September 2026, Anda tidak lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi

lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Topik ini merinci masalah umum yang mungkin Anda alami dengan keamanan App Mesh.

Tidak dapat terhubung ke layanan virtual backend dengan kebijakan klien TLS

Gejala

Saat menambahkan kebijakan klien TLS ke backend layanan virtual di node virtual, konektivitas ke backend tersebut gagal. Saat mencoba mengirim lalu lintas ke layanan backend, permintaan gagal dengan kode HTTP 503 respons dan pesan kesalahan: `upstream connect error or disconnect/reset before headers. reset reason: connection failure`

Resolusi

Untuk menentukan akar penyebab masalah, sebaiknya gunakan log proses proxy Envoy untuk membantu Anda mendiagnosis masalah. Untuk informasi selengkapnya, lihat [Aktifkan logging debug Envoy di lingkungan pra-produksi](#). Gunakan daftar berikut untuk menentukan penyebab kegagalan koneksi:

- Pastikan konektivitas ke backend berhasil dengan mengesampingkan kesalahan yang disebutkan di [Tidak dapat terhubung ke backend layanan virtual](#)
- Di log proses Envoy, cari kesalahan berikut (dicatat pada tingkat debug).

```
TLS error: 268435581:SSL routines:OPENSSL_internal:CERTIFICATE_VERIFY_FAILED
```

Kesalahan ini disebabkan oleh satu atau lebih alasan berikut:

- Sertifikat tidak ditandatangani oleh salah satu otoritas sertifikat yang ditentukan dalam paket kepercayaan kebijakan klien TLS.
- Sertifikat tidak lagi valid (kedaluwarsa).
- Nama Alternatif Subjek (SAN) tidak cocok dengan nama host DNS yang diminta.
- Pastikan bahwa sertifikat yang ditawarkan oleh layanan backend valid, yang ditandatangani oleh salah satu otoritas sertifikat dalam paket kepercayaan kebijakan klien TLS Anda, dan memenuhi kriteria yang ditentukan. [Keamanan Lapisan Pengangkutan \(TLS\)](#)

- Jika kesalahan yang Anda terima seperti yang di bawah ini, maka itu berarti permintaan tersebut melewati proxy Utusan dan mencapai aplikasi secara langsung. Saat mengirim lalu lintas, statistik di Envoy tidak berubah yang menunjukkan bahwa Utusan tidak berada di jalur untuk mendekripsi lalu lintas. Dalam konfigurasi proxy node virtual, pastikan AppPorts berisi nilai yang benar yang didengarkan aplikasi.

```
upstream connect error or disconnect/reset before headers. reset reason:
connection failure, transport failure reason: TLS error: 268435703:SSL
routines:OPENSSL_internal:WRONG_VERSION_NUMBER
```

Jika masalah Anda masih belum teratasi, pertimbangkan untuk membuka [GitHub masalah](#) atau hubungi [AWS Support](#). Jika Anda yakin telah menemukan kerentanan keamanan atau memiliki pertanyaan tentang keamanan App Mesh, lihat pedoman [pelaporan AWS kerentanan](#).

Tidak dapat terhubung ke layanan virtual backend saat aplikasi berasal dari TLS

Gejala

Saat memulai sesi TLS dari aplikasi, bukan dari proxy Envoy, konektivitas ke layanan virtual backend gagal.

Resolusi

Ini adalah masalah yang diketahui. Untuk informasi selengkapnya, lihat [Permintaan Fitur: negosiasi TLS antara aplikasi hilir dan masalah proksi hulu](#). GitHub Di App Mesh, originasi TLS saat ini didukung dari proxy Envoy tetapi tidak dari aplikasi. Untuk menggunakan dukungan originasi TLS di Utusan, nonaktifkan originasi TLS dalam aplikasi. Hal ini memungkinkan Utusan untuk membaca header permintaan keluar dan meneruskan permintaan ke tujuan yang sesuai melalui sesi TLS. Untuk informasi selengkapnya, lihat [Keamanan Lapisan Pengangkutan \(TLS\)](#).

Jika masalah Anda masih belum teratasi, pertimbangkan untuk membuka [GitHub masalah](#) atau hubungi [AWS Support](#). Jika Anda yakin telah menemukan kerentanan keamanan atau memiliki pertanyaan tentang keamanan App Mesh, lihat pedoman [pelaporan AWS kerentanan](#).

Tidak dapat menegaskan bahwa konektivitas antara proxy Utusan menggunakan TLS

Gejala

Aplikasi Anda telah mengaktifkan penghentian TLS pada node virtual atau pendengar gateway virtual, atau orinasi TLS pada kebijakan klien TLS backend, tetapi Anda tidak dapat menegaskan bahwa konektivitas antara proxy Envoy terjadi selama sesi yang dinegosiasikan TLS.

Resolusi

Langkah-langkah yang ditentukan dalam resolusi ini menggunakan antarmuka administrasi Utusan dan statistik Utusan. Untuk bantuan mengonfigurasi ini, lihat [Aktifkan antarmuka administrasi proxy Envoy](#) dan [Aktifkan integrasi Envoy DogStats D untuk pembongkaran metrik](#). Contoh statistik berikut menggunakan antarmuka administrasi untuk kesederhanaan.

- Untuk proxy Utusan yang melakukan penghentian TLS:
 - Pastikan bahwa sertifikat TLS telah di-bootstrap dalam konfigurasi Envoy dengan perintah berikut.

```
curl http://my-app.default.svc.cluster.local:9901/certs
```

Dalam output yang dikembalikan, Anda harus melihat setidaknya satu entri di bawah `certificates[].cert_chain` untuk sertifikat yang digunakan dalam penghentian TLS.

- Pastikan bahwa jumlah koneksi masuk yang berhasil ke pendengar proxy sama persis dengan jumlah jabat tangan SSL ditambah jumlah sesi SSL yang digunakan kembali, seperti yang ditunjukkan oleh contoh perintah dan output berikut.

```
curl -s http://my-app.default.svc.cluster.local:9901/stats | grep  
"listener.0.0.0.0_15000" | grep downstream_cx_total  
listener.0.0.0.0_15000.downstream_cx_total: 11  
curl -s http://my-app.default.svc.cluster.local:9901/stats | grep  
"listener.0.0.0.0_15000" | grep ssl.connection_error  
listener.0.0.0.0_15000.ssl.connection_error: 1  
curl -s http://my-app.default.svc.cluster.local:9901/stats | grep  
"listener.0.0.0.0_15000" | grep ssl.handshake  
listener.0.0.0.0_15000.ssl.handshake: 9  
curl -s http://my-app.default.svc.cluster.local:9901/stats | grep  
"listener.0.0.0.0_15000" | grep ssl.session_reused
```

```
listener.0.0.0.0_15000.ssl.session_reused: 1
# Total CX (11) - SSL Connection Errors (1) == SSL Handshakes (9) + SSL Sessions
Re-used (1)
```

- Untuk proxy Envoy yang melakukan originasi TLS:
 - Pastikan bahwa toko kepercayaan TLS telah di-bootstrap dalam konfigurasi Envoy dengan perintah berikut.

```
curl http://my-app.default.svc.cluster.local:9901/certs
```

Anda harus melihat setidaknya satu entri di bawah `certificates[].ca_certs` untuk sertifikat yang digunakan dalam memvalidasi sertifikat backend selama originasi TLS.

- Pastikan bahwa jumlah koneksi keluar yang berhasil ke cluster backend persis sama dengan jumlah jabat tangan SSL ditambah jumlah sesi SSL yang digunakan kembali, seperti yang ditunjukkan oleh contoh perintah dan output berikut.

```
curl -s http://my-app.default.svc.cluster.local:9901/stats | grep "virtual-node-name" | grep upstream_cx_total
cluster.cds_egress_mesh-name_virtual-node-name_protocol_port.upstream_cx_total: 11
curl -s http://my-app.default.svc.cluster.local:9901/stats | grep "virtual-node-name" | grep ssl.connection_error
cluster.cds_egress_mesh-name_virtual-node-name_protocol_port.ssl.connection_error:
1
curl -s http://my-app.default.svc.cluster.local:9901/stats | grep "virtual-node-name" | grep ssl.handshake
cluster.cds_egress_mesh-name_virtual-node-name_protocol_port.ssl.handshake: 9
curl -s http://my-app.default.svc.cluster.local:9901/stats | grep "virtual-node-name" | grep ssl.session_reused
cluster.cds_egress_mesh-name_virtual-node-name_protocol_port.ssl.session_reused: 1
# Total CX (11) - SSL Connection Errors (1) == SSL Handshakes (9) + SSL Sessions
Re-used (1)
```

Jika masalah Anda masih belum teratasi, pertimbangkan untuk membuka [GitHub masalah](#) atau hubungi [AWS Support](#). Jika Anda yakin telah menemukan kerentanan keamanan atau memiliki pertanyaan tentang keamanan App Mesh, lihat pedoman [pelaporan AWS kerentanan](#).

Memecahkan Masalah TLS dengan Elastic Load Balancing

Gejala

Saat mencoba mengonfigurasi Application Load Balancer atau Network Load Balancer untuk mengenkripsi lalu lintas ke node virtual, pemeriksaan kesehatan konektivitas dan penyeimbang beban dapat gagal.

Resolusi

Untuk menentukan akar penyebab masalah, Anda perlu memeriksa yang berikut ini:

- Untuk proxy Envoy yang melakukan penghentian TLS, Anda harus mengesampingkan kesalahan konfigurasi apa pun. Ikuti langkah-langkah yang diberikan di atas di [Tidak dapat terhubung ke layanan virtual backend dengan kebijakan klien TLS](#).
- Untuk penyeimbang beban, Anda perlu melihat konfigurasi TargetGroup:
 - Pastikan TargetGroup port cocok dengan port pendengar yang ditentukan node virtual.
 - Untuk Application Load Balancers yang berasal dari koneksi TLS melalui HTTP ke layanan Anda, pastikan bahwa TargetGroup protokol diatur ke HTTPS. Jika pemeriksaan kesehatan sedang digunakan, pastikan itu HealthCheckProtocol diatur ke HTTPS.
 - Untuk Network Load Balancer yang berasal dari koneksi TLS melalui TCP ke layanan Anda, pastikan bahwa protokol diatur ke TargetGroup. TLS. Jika pemeriksaan kesehatan sedang digunakan, pastikan itu HealthCheckProtocol diatur ke TCP.

Note

Setiap pembaruan yang TargetGroup memerlukan perubahan TargetGroup nama.

Dengan konfigurasi ini dengan benar, penyeimbang beban Anda harus menyediakan koneksi aman ke layanan Anda menggunakan sertifikat yang diberikan kepada proxy Envoy.

Jika masalah Anda masih belum teratasi, pertimbangkan untuk membuka [GitHub masalah](#) atau hubungi [AWS Support](#). Jika Anda yakin telah menemukan kerentanan keamanan atau memiliki pertanyaan tentang keamanan App Mesh, lihat pedoman [pelaporan AWS kerentanan](#).

Pemecahan masalah App Mesh Kubernetes

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk AWS App Mesh. Setelah 30 September 2026, Anda tidak lagi dapat

mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Topik ini merinci masalah umum yang mungkin Anda alami saat menggunakan App Mesh dengan Kubernetes.

Sumber daya App Mesh yang dibuat di Kubernetes tidak dapat ditemukan di App Mesh

Gejala

Anda telah membuat resource App Mesh menggunakan definisi sumber daya kustom Kubernetes (CRD), tetapi resource yang Anda buat tidak terlihat di App Mesh saat Anda menggunakan or. Konsol Manajemen AWS APIs

Resolusi

Kemungkinan penyebabnya adalah kesalahan pada pengontrol Kubernetes untuk App Mesh. Untuk informasi selengkapnya, lihat [Pemecahan Masalah](#) di GitHub Periksa log pengontrol untuk setiap kesalahan atau peringatan yang menunjukkan bahwa pengontrol tidak dapat membuat sumber daya apa pun.

```
kubectl logs -n appmesh-system -f \  
$(kubectl get pods -n appmesh-system -o name | grep controller)
```

Jika masalah Anda masih belum teratasi, pertimbangkan untuk membuka [GitHub masalah](#) atau hubungi [AWS Support](#).

Pod gagal memeriksa kesiapan dan keaktifan setelah envoy sidecar disuntikkan

Gejala

Pod untuk aplikasi Anda sebelumnya berhasil berjalan, tetapi setelah sespan Envoy disuntikkan ke dalam pod, pemeriksaan kesiapan dan keaktifan mulai gagal.

Resolusi

Pastikan bahwa wadah Envoy yang disuntikkan ke dalam pod telah di-bootstrap dengan layanan manajemen Envoy App Mesh. Anda dapat memverifikasi kesalahan apa pun dengan mereferensikan kode kesalahan di [Utusan terputus dari layanan manajemen App Mesh Envoy dengan teks kesalahan](#). Anda dapat menggunakan perintah berikut untuk memeriksa log Envoy untuk pod yang relevan.

```
kubectl logs -n appmesh-system -f \  
  $(kubectl get pods -n appmesh-system -o name | grep controller) \  
  | grep "gRPC config stream closed"
```

Jika masalah Anda masih belum teratasi, pertimbangkan untuk membuka [GitHub masalah](#) atau hubungi [AWS Support](#).

Pod tidak mendaftar atau membatalkan pendaftaran sebagai instance AWS Cloud Map

Gejala

Pod Kubernetes Anda tidak terdaftar atau tidak terdaftar AWS Cloud Map sebagai bagian dari siklus hidupnya. Sebuah pod dapat memulai dengan sukses dan siap untuk melayani lalu lintas, tetapi tidak menerima apa pun. Ketika sebuah pod dihentikan, klien mungkin masih mempertahankan alamat IP-nya dan mencoba mengirim lalu lintas ke sana, gagal.

Resolusi

Ini adalah masalah yang diketahui. Untuk informasi selengkapnya, lihat [Pod don't get auto registered/deregistered di Kubernetes](#) dengan masalah. AWS Cloud Map GitHub Karena hubungan antara pod, node virtual App Mesh, dan AWS Cloud Map sumber daya, [pengontrol App Mesh untuk Kubernetes](#) dapat menjadi tidak sinkron dan kehilangan sumber daya. Misalnya, hal ini dapat terjadi jika sumber daya node virtual dihapus dari Kubernetes sebelum menghentikan pod yang terkait.

Untuk mengurangi masalah ini:

- Pastikan Anda menjalankan versi terbaru dari pengontrol App Mesh untuk Kubernetes.
- Pastikan bahwa AWS Cloud Map namespaceName dan serviceName benar dalam definisi node virtual Anda.
- Pastikan Anda menghapus semua Pod terkait sebelum menghapus definisi node virtual Anda. Jika Anda memerlukan bantuan untuk mengidentifikasi pod mana yang terkait dengan node virtual, lihat [Tidak dapat menentukan lokasi pod untuk resource App Mesh berjalan](#).

- Jika masalah Anda berlanjut, jalankan perintah berikut untuk memeriksa log pengontrol Anda untuk kesalahan yang dapat membantu mengungkapkan masalah mendasar.

```
kubectl logs -n appmesh-system \  
$(kubectl get pods -n appmesh-system -o name | grep appmesh-controller)
```

- Pertimbangkan untuk menggunakan perintah berikut untuk me-restart pod pengontrol Anda. Ini dapat memperbaiki masalah sinkronisasi.

```
kubectl delete -n appmesh-system \  
$(kubectl get pods -n appmesh-system -o name | grep appmesh-controller)
```

Jika masalah Anda masih belum teratasi, pertimbangkan untuk membuka [GitHub masalah](#) atau hubungi [AWS Support](#).

Tidak dapat menentukan lokasi pod untuk resource App Mesh berjalan

Gejala

Ketika Anda menjalankan App Mesh di klaster Kubernetes, operator tidak dapat menentukan di mana beban kerja, atau pod, berjalan untuk resource App Mesh tertentu.

Resolusi

Sumber daya pod Kubernetes dianotasi dengan mesh dan node virtual yang terkait dengannya. Anda dapat menanyakan pod mana yang sedang berjalan untuk nama node virtual tertentu dengan perintah berikut.

```
kubectl get pods --all-namespaces -o json | \  
jq '.items[] | { metadata } | select(.metadata.annotations."appmesh.k8s.aws/  
virtualNode" == "virtual-node-name")'
```

Jika masalah Anda masih belum teratasi, pertimbangkan untuk membuka [GitHub masalah](#) atau hubungi [AWS Support](#).

Tidak dapat menentukan sumber daya App Mesh apa yang dijalankan pod

Gejala

Saat menjalankan App Mesh di kluster Kubernetes, operator tidak dapat menentukan resource App Mesh apa yang dijalankan pod tertentu.

Resolusi

Sumber daya pod Kubernetes dianotasi dengan mesh dan node virtual yang terkait dengannya. Anda dapat menampilkan nama mesh dan node virtual dengan menanyakan pod secara langsung menggunakan perintah berikut.

```
kubectl get pod pod-name -n namespace -o json | \
jq '{ "mesh": .metadata.annotations."appmesh.k8s.aws/mesh",
"virtualNode": .metadata.annotations."appmesh.k8s.aws/virtualNode" }'
```

Jika masalah Anda masih belum teratasi, pertimbangkan untuk membuka [GitHub masalah](#) atau hubungi [AWS Support](#).

Utusan Klien tidak dapat berkomunikasi dengan Layanan Manajemen Utusan App Mesh dengan dinonaktifkan IMDSv1

Gejala

Saat IMDSv1 dinonaktifkan, Utusan klien tidak dapat berkomunikasi dengan bidang kontrol App Mesh (Layanan Manajemen Utusan). IMDSv2 dukungan tidak tersedia pada versi App Mesh Envoy sebelumnya. `v1.24.0.0-prod`

Resolusi

Untuk mengatasi masalah ini, Anda dapat melakukan salah satu dari tiga hal ini.

- Tingkatkan ke versi App Mesh Envoy `v1.24.0.0-prod` atau yang lebih baru, yang memiliki IMDSv2 dukungan.
- Aktifkan kembali IMDSv1 pada Instance tempat Utusan berjalan. Untuk petunjuk tentang memulihkan IMDSv1, lihat [Mengkonfigurasi opsi metadata instance](#).
- Jika layanan Anda berjalan di Amazon EKS, disarankan untuk menggunakan peran IAM untuk akun layanan (IRSA) untuk mengambil kredensial. Untuk petunjuk mengaktifkan IRSA, lihat [peran IAM untuk akun layanan](#).

Jika masalah Anda masih belum teratasi, pertimbangkan untuk membuka [GitHub masalah](#) atau hubungi [AWS Support](#).

IRSA tidak berfungsi pada wadah aplikasi saat App Mesh diaktifkan dan Utusan disuntikkan

Gejala

Saat App Mesh diaktifkan pada kluster Amazon EKS dengan bantuan pengontrol App Mesh untuk Amazon EKS, Utusan dan `proxyinit` kontainer akan disuntikkan ke dalam pod aplikasi. Aplikasi tidak dapat mengasumsikan IRSA dan sebagai gantinya mengasumsikan `node role`. Ketika kita menjelaskan rincian pod, kita kemudian melihat bahwa variabel `AWS_WEB_IDENTITY_TOKEN_FILE` atau `AWS_ROLE_ARN` lingkungan tidak termasuk dalam wadah aplikasi.

Resolusi

Jika salah satu `AWS_WEB_IDENTITY_TOKEN_FILE` atau variabel `AWS_ROLE_ARN` lingkungan didefinisikan, maka webhook akan melewati pod. Jangan berikan salah satu dari variabel-variabel ini dan webhook akan menyuntikkannya untuk Anda.

```
reservedKeys := map[string]string{
    "AWS_ROLE_ARN": "",
    "AWS_WEB_IDENTITY_TOKEN_FILE": "",
}
...
for _, env := range container.Env {
    if _, ok := reservedKeys[env.Name]; ok {
        reservedKeysDefined = true
    }
}
```

Jika masalah Anda masih belum teratasi, pertimbangkan untuk membuka [GitHub masalah](#) atau hubungi [AWS Support](#).

Kuota layanan App Mesh

⚠ Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk AWS App Mesh. Setelah 30 September 2026, Anda tidak lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

AWS App Mesh telah terintegrasi dengan Service Quotas, sebuah AWS layanan yang memungkinkan Anda untuk melihat dan mengelola kuota Anda dari lokasi pusat. Kuota layanan juga disebut sebagai batas. Untuk informasi selengkapnya, lihat [Apa itu Service Quotas?](#) di Panduan Pengguna Service Quotas.

Service Quotas memudahkan untuk mencari nilai semua kuota layanan App Mesh.

Untuk melihat kuota layanan App Mesh menggunakan Konsol Manajemen AWS

1. Buka konsol Service Quotas di <https://console.aws.amazon.com/servicequotas/>.
2. Di panel navigasi, pilih Layanan AWS .
3. Dari daftar Layanan AWS , cari dan pilih AWS App Mesh.

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 kuota layanan, seperti deskripsi, pilih nama kuota.

Untuk meminta peningkatan kuota, lihat [Meminta Peningkatan Kuota](#) dalam Panduan Pengguna Service Quotas.

Untuk melihat kuota layanan App Mesh menggunakan AWS CLI

Jalankan perintah berikut.

```
aws service-quotas list-aws-default-service-quotas \
  --query 'Quotas[*]'.
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
```

```
--service-code appmesh \  
--output table
```

Untuk bekerja lebih banyak dengan kuota layanan menggunakan AWS CLI, lihat Referensi Perintah [Service AWS CLI Quotas](#).

Riwayat dokumen untuk App Mesh

Important

Pemberitahuan akhir dukungan: Pada 30 September 2026, AWS akan menghentikan dukungan untuk AWS App Mesh. Setelah 30 September 2026, Anda tidak akan lagi dapat mengakses AWS App Mesh konsol atau AWS App Mesh sumber daya. Untuk informasi lebih lanjut, kunjungi posting blog ini [Migrasi dari AWS App Mesh ke Amazon ECS Service Connect](#).

Tabel berikut menjelaskan pembaruan utama dan fitur baru untuk Panduan AWS App Mesh Pengguna. Kami juga rutin memperbarui dokumentasi untuk menjawab umpan balik yang Anda kirimkan kepada kami.

Perubahan	Deskripsi	Tanggal
AWSAppMeshFullAccess Kebijakan yang diperbarui	Diperbarui <code>AWSAppMeshFullAccess</code> untuk memungkinkan akses ke <code>TagResource</code> dan <code>UntagResource</code> APIs.	April 24, 2024
CloudTrail dokumentasi integrasi diperbarui	Dokumentasi yang menjelaskan integrasi App Mesh CloudTrail dengan aktivitas API log telah diperbarui.	Maret 28, 2024
Kebijakan yang diperbarui	Diperbarui <code>AWSServiceRoleForAppMesh</code> dan <code>AWSAppMeshServiceRolePolicy</code> untuk memungkinkan akses ke <code>AWS Cloud Map DiscoverInstancesRevision</code> API.	12 Oktober 2023

Dukungan kebijakan titik akhir VPC untuk App Mesh	App Mesh sekarang mendukung kebijakan titik akhir VPC.	11 Mei 2023
Beberapa pendengar untuk App Mesh	App Mesh sekarang mendukung banyak pendengar.	18 Agustus 2022
IPv6 untuk App Mesh	App Mesh sekarang mendukung IPv6.	Mei 18, 2022
CloudTrail dukungan logging untuk Layanan Manajemen Utusan App Mesh	App Mesh sekarang mendukung dukungan CloudTrail logging untuk App Mesh Envoy Management Service.	18 Maret 2022
Agen App Mesh untuk Utusan	App Mesh sekarang mendukung Agen untuk Utusan.	25 Februari 2022
Beberapa pendengar untuk App Mesh	(Hanya Saluran Pratinjau App Mesh) Anda dapat menerapkan beberapa pendengar untuk App Mesh.	23 November 2021
ARM64 dukungan untuk App Mesh	App Mesh sekarang mendukung ARM64.	November 19, 2021
Ekstensi metrik untuk App Mesh	Anda dapat menerapkan ekstensi metrik untuk App Mesh.	29 Oktober 2021
Menerapkan peningkatan lalu lintas masuk	Anda dapat menerapkan nama host dan pencocokan header dan penulisan ulang untuk nama dan jalur host.	14 Juni 2021

Menerapkan otentikasi TLS timbal balik	Anda dapat menerapkan otentikasi TLS timbal balik.	4 Februari 2021
Peluncuran wilayah di af-south-1	App Mesh sekarang tersedia di Wilayah af-south-1.	22 Januari 2021
Menerapkan otentikasi TLS timbal balik	(Hanya Saluran Pratinjau App Mesh) Anda dapat menerapkan otentikasi TLS timbal balik.	23 November 2020
Menerapkan penyatuan koneksi untuk pendengar gateway virtual	Anda dapat menerapkan penyatuan koneksi untuk pendengar gateway virtual.	5 November 2020
Menerapkan pengumpulan koneksi dan deteksi outlier untuk pendengar simpul virtual	Anda dapat menerapkan penyatuan koneksi dan deteksi outlier untuk pendengar simpul virtual.	5 November 2020
Peluncuran wilayah di eu-south-1	App Mesh sekarang tersedia di Wilayah eu-south-1.	21 Oktober 2020
Menerapkan penyatuan koneksi untuk pendengar gateway virtual	(Hanya Saluran Pratinjau App Mesh) Anda dapat menerapkan penyatuan koneksi untuk pendengar gateway virtual.	Senin, 28 September 2020
Menerapkan pengumpulan koneksi dan deteksi outlier untuk pendengar simpul virtual	(Hanya Saluran Pratinjau App Mesh) Anda dapat menerapkan pengumpulan koneksi dan deteksi outlier untuk pendengar simpul virtual.	Senin, 28 September 2020
Buat gateway virtual dan rute gateway untuk mesh inbound	Aktifkan sumber daya yang berada di luar mesh untuk berkomunikasi dengan sumber daya yang ada di dalam mesh.	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. Pengendali juga secara otomatis menyuntikkan proksi Envoy dan kontainer init ke pod yang Anda deploy.	18 Juni 2020
Tambahkan nilai batas waktu ke pendengar dan rute node virtual	Anda dapat menambahkan nilai batas waktu ke pendengar dan rute node virtual.	18 Juni 2020
Tambahkan nilai batas waktu ke pendengar simpul virtual	(Hanya Saluran Pratinjau App Mesh) Anda dapat menambahkan nilai batas waktu ke pendengar simpul virtual.	29 Mei 2020
Buat gateway virtual untuk mesh inbound	(Hanya Saluran Pratinjau App Mesh) Aktifkan sumber daya di luar mesh untuk berkomunikasi dengan sumber daya di dalam mesh.	8 April 2020
Enkripsi TLS	(Hanya Saluran Pratinjau App Mesh) Gunakan sertifikat dari AWS Private Certificate Authority atau otoritas sertifikat Anda sendiri untuk mengenkripsi komunikasi antar node virtual menggunakan TLS.	17 Januari 2020

Bagikan mesh dengan akun lain	(Hanya Saluran Pratinjau App Mesh) Anda dapat berbagi mesh dengan akun lain. Sumber daya yang dibuat oleh akun apa pun dapat berkomunikasi dengan sumber daya lain di mesh.	17 Januari 2020
Tambahkan nilai batas waktu ke rute	(Hanya Saluran Pratinjau App Mesh) Anda dapat menambahkan nilai batas waktu ke rute.	17 Januari 2020
Membuat proxy App Mesh di AWS Outpost	Anda dapat membuat proxy App Mesh Envoy di Outpost. AWS	3 Desember 2019
Dukungan HTTP/2 dan gRPC untuk rute, router virtual, dan node virtual	Anda dapat merutekan lalu lintas yang menggunakan protokol HTTP/2 dan gRPC. Anda juga dapat menambahkan pendengar untuk protokol ini ke node virtual dan router virtual.	Oktober 25, 2019
Coba lagi kebijakan	Kebijakan coba lagi memungkinkan klien untuk melindungi diri dari kegagalan jaringan intermiten atau kegagalan sisi server intermiten. Anda dapat menambahkan logika coba lagi ke rute.	10 September 2019
Enkripsi TLS	(Hanya Saluran Pratinjau App Mesh) Enkripsi komunikasi antara node virtual menggunakan TLS.	6 September 2019

Perutean berbasis header HTTP	Rute lalu lintas berdasarkan keberadaan dan nilai header HTTP dalam permintaan.	15 Agustus 2019
Ketersediaan Saluran Pratinjau App Mesh	Saluran Pratinjau App Mesh adalah varian berbeda dari layanan App Mesh. Saluran Pratinjau memperlihatkan fitur yang akan datang untuk Anda coba saat dikembangkan. Saat Anda menggunakan fitur di Saluran Pratinjau, Anda dapat memberikan umpan balik melalui GitHub untuk membentuk perilaku fitur.	19 Juli 2019
Antarmuka App Mesh Titik Akhir VPC (AWS PrivateLink)	Tingkatkan postur keamanan VPC Anda dengan mengonfigurasi App Mesh untuk menggunakan titik akhir VPC antarmuka. Endpoint antarmuka didukung oleh AWS PrivateLink, teknologi yang memungkinkan Anda mengakses App Mesh secara pribadi APIs dengan menggunakan alamat IP pribadi. PrivateLink membatasi semua lalu lintas jaringan antara VPC dan App Mesh Anda ke jaringan Amazon.	14 Juni 2019

Ditambahkan AWS Cloud Map sebagai metode penemuan layanan node virtual	Anda dapat menentukan DNS atau AWS Cloud Map sebagai metode penemuan layanan node virtual. Untuk digunakan AWS Cloud Map untuk penemuan layanan, akun Anda harus memiliki peran terkait layanan App Mesh.	13 Juni 2019
Buat resource App Mesh secara otomatis di Kubernetes	Buat resource App Mesh dan tambahkan image container sidecar App Mesh ke deployment Kubernetes Anda secara otomatis saat Anda membuat resource di Kubernetes.	11 Juni 2019
Ketersediaan Umum App Mesh	Layanan App Mesh sekarang umumnya tersedia untuk penggunaan produksi.	27 Maret 2019
Pembaruan App Mesh API	App Mesh APIs diperbarui untuk meningkatkan kegunaan. Untuk informasi selengkapnya, lihat [BUG] Rute ke Target Node Virtual dengan Lubang Hitam Port yang Tidak Cocokkan .	7 Maret 2019
Rilis awal App Mesh	Dokumentasi awal untuk pratinjau layanan publik	28 November, 2018

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