Panduan Developer

AWS App Runner

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

AWS App Runner: Panduan Developer

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

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

Table of Contents

Apa itu AWS App Runner? ... 1
Untuk siapa App Runner? ... 1
Mengakses Pelari Aplikasi ... 1
Harga untuk App Runner ... 2
Apa selanjutnya ... 2
Pengaturan ... 3
Mendaftar untuk Akun AWS ... 3
Buat pengguna dengan akses administratif ... 3
Memberikan akses programatis ... 5
Apa selanjutnya ... 6
Memulai ... 7
Prasyarat ... 7
Langkah 1: Buat layanan App Runner ... 9
Langkah 2: Ubah kode layanan Anda ... 19
Langkah 3: Buat perubahan konfigurasi ... 20
Langkah 4: Lihat log untuk layanan Anda ... 22
Langkah 5: Bersihkan ... 25
Apa selanjutnya ... 25
Arsitektur dan konsep ... 27
Konsep App Runner ... 28
Konfigurasi yang didukung App Runner ... 29
Sumber daya App Runner ... 30
Kuota sumber daya App Runner ... 32
Layanan berbasis gambar ... 35
Penyedia repositori gambar ... 35
Menggunakan gambar yang disimpan di Amazon ECR di akun Anda AWS ... 36
Menggunakan gambar yang disimpan di Amazon ECR di akun yang berbeda AWS ... 36
Menggunakan gambar yang disimpan di Amazon ECR Public ... 37
Contoh gambar ... 38
Layanan berbasis kode ... 39
Penyedia repositori kode sumber ... 40
Menerapkan dari penyedia repositori kode sumber Anda ... 40
Direktori sumber ... 41
Platform terkelola App Runner ... 42

Versi runtime terkelola dan build App Runner ... 42
Selengkapnya tentang build dan migrasi App Runner ... 43
Platform Python ... 47
Konfigurasi runtime Python ... 49
Callout untuk versi runtime tertentu ... 49
Contoh runtime Python ... 50
Informasi rilis ... 55
Platform Node.js ... 56
Konfigurasi runtime Node.js ... 57
Callout untuk versi runtime tertentu ... 60
Contoh runtime Node.js ... 60
Informasi rilis ... 64
Platform Java ... 67
Konfigurasi runtime Java ... 68
Contoh runtime Java ... 69
Informasi rilis ... 73
.NET ... 75
Konfigurasi runtime .NET ... 76
Contoh runtime .NET ... 76
Informasi rilis ... 79
Platform PHP ... 80
Konfigurasi runtime PHP ... 82
Kompatibilitas ... 82
Contoh runtime PHP ... 84
Informasi rilis ... 92
Platform Ruby ... 94
Konfigurasi runtime Ruby ... 95
Contoh runtime Ruby ... 95
Informasi rilis ... 97
Pergi platform ... 98
Konfigurasi runtime Go ... 99
Contoh runtime Go ... 100
Informasi rilis ... 102
Mengembangkan untuk App Runner ... 103
Informasi runtime ... 103
Pedoman pengembangan kode ... 105

Konsol Pelari Aplikasi ... 106
Tata letak konsol keseluruhan ... 106
Halaman Layanan ... 107
Halaman dasbor layanan ... 107
Halaman Akun Terhubung ... 109
Halaman konfigurasi penskalaan Otomatis ... 109
Mengelola layanan Anda ... 111
Pembuatan ... 111
Prasyarat ... 112
Membuat layanan ... 112
Membangun kembali layanan yang gagal ... 128
Membangun kembali layanan App Runner yang gagal menggunakan konsol App Runner ... 128
Membangun kembali layanan App Runner yang gagal menggunakan App Runner API atau AWS CLI ... 129
Deployment ... 130
Metode deployment ... 130
Penyebaran manual ... 132
Konfigurasi ... 134
Konfigurasikan layanan Anda menggunakan App Runner API atau AWS CLI ... 135
Konfigurasikan layanan Anda menggunakan konsol App Runner ... 136
Konfigurasikan layanan Anda menggunakan file konfigurasi App Runner ... 138
Konfigurasi observabilitas ... 138
Sumber daya konfigurasi ... 140
Konfigurasi pemeriksaan kesehatan ... 142
Koneksi ... 144
Kelola koneksi ... 144
Penskalaan otomatis ... 146
Mengelola penskalaan otomatis untuk suatu layanan ... 148
Mengelola sumber daya konfigurasi penskalaan otomatis ... 150
Nama domain khusus ... 157
Kaitkan (tautan) domain khusus ke layanan Anda ... 158
Putuskan (putuskan tautan) domain khusus ... 161
Mengelola domain kustom ... 162
Konfigurasikan catatan alias Amazon Route 53 ... 170
Menjeda/melanjutkan ... 172
Menjeda dan menghapus dibandingkan ... 173

Saat layanan Anda dijeda ... 173
Jeda dan lanjutkan layanan Anda ... 174
Penghapusan ... 176
Menjeda dan menghapus dibandingkan ... 176
Apa yang dihapus oleh App Runner? ... 177
Hapus layanan Anda ... 177
Referensi variabel Lingkungan ... 179
Mereferensikan data sensitif sebagai variabel lingkungan ... 179
Pertimbangan ... 181
Izin ... 181
Kelola variabel lingkungan ... 183
Konsol Pelari Aplikasi ... 183
API Pelari Aplikasi atau AWS CLI ... 185
Jaringan ... 191
Terminologi ... 191
Ketentuan Umum ... 191
Istilah khusus untuk mengonfigurasi lalu lintas keluar ... 192
Ketentuan khusus untuk mengonfigurasi lalu lintas masuk ... 192
Lalu lintas masuk ... 193
Header ... 194
Aktifkan titik akhir Pribadi ... 194
Aktifkan IPv6 untuk titik akhir publik App Runner ... 207
Lalu lintas keluar ... 212
Konektor VPC ... 213
Subnet ... 214
Grup keamanan ... 214
Kelola akses VPC ... 215
Observabilitas ... 221
Aktifitas ... 221
Lacak aktivitas layanan App Runner ... 221
Log (CloudWatch Log) ... 222
Grup dan aliran log Pelari Aplikasi ... 223
Melihat log App Runner di konsol ... 224
Metrik () CloudWatch ... 227
Metrik Pelari Aplikasi ... 227
Melihat metrik App Runner di konsol ... 229

Penanganan acara (EventBridge) ... 231
Membuat EventBridge aturan untuk bertindak pada acara App Runner ... 232
Contoh acara App Runner ... 232
Contoh pola acara App Runner ... 234
Referensi acara App Runner ... 235
Tindakan API (CloudTrail) ... 236
Informasi Pelari Aplikasi di CloudTrail ... 237
Memahami entri file log App Runner ... 238
Penelusuran (X-Ray) ... 241
Instrumen aplikasi Anda untuk melacak ... 242
Menambahkan izin X-Ray ke peran instance layanan App Runner ... 245
Aktifkan penelusuran X-Ray untuk layanan App Runner ... 246
Melihat data penelusuran X-Ray untuk layanan App Runner ... 246
AWS WAF web ACL ... 247
Alur permintaan web yang masuk ... 247
Mengaitkan web WAF ACLs ke layanan App Runner Anda ... 248
Pertimbangan ... 249
Izin ... 250
Kelola web ACLs ... 251
Konsol Pelari Aplikasi ... 251
AWS CLI ... 255
Menguji dan mencatat AWS WAF web ACLs ... 260
File konfigurasi App Runner ... 262
Contoh ... 263
Contoh file konfigurasi ... 263
Referensi ... 266
Ikhtisar struktur ... 266
Bagian atas ... 267
Membangun bagian ... 267
Jalankan bagian ... 269
API Pelari Aplikasi ... 274
Menggunakan AWS CLI untuk bekerja dengan App Runner ... 274
Menggunakan AWS CloudShell ... 274
Memperoleh izin IAM untuk AWS CloudShell ... 275
Berinteraksi dengan App Runner menggunakan AWS CloudShell ... 276
Memverifikasi layanan App Runner Anda menggunakan AWS CloudShell ... 279

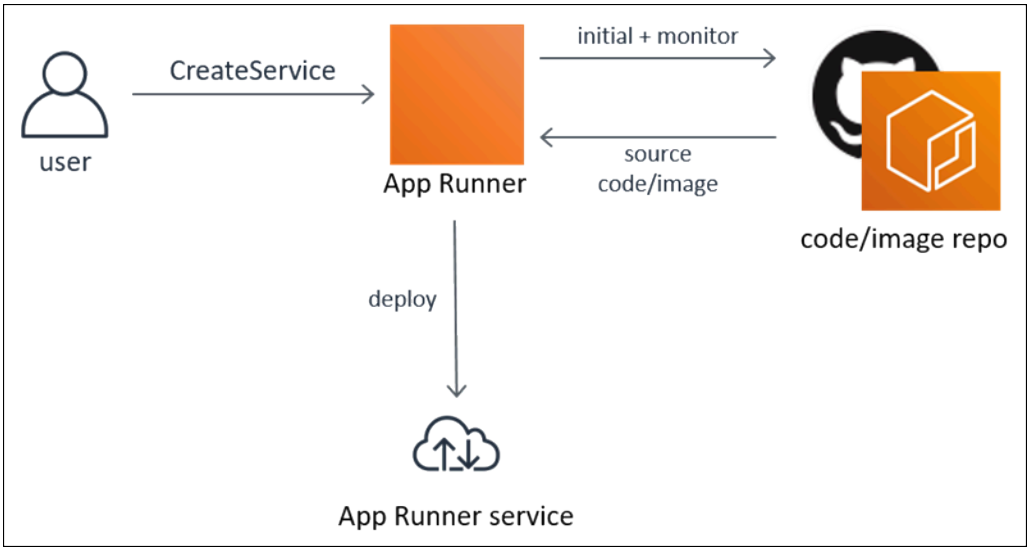
Pemecahan Masalah ... 280
Gagal membuat layanan ... 280
Nama domain kustom ... 281
Mendapatkan kesalahan Buat Gagal untuk domain kustom ... 282
Mendapatkan validasi sertifikat DNS kesalahan tertunda untuk domain kustom ... 283
Perintah pemecahan masalah dasar ... 283
Perpanjangan sertifikat domain kustom ... 284
Minta kesalahan perutean ... 285
404 Kesalahan tidak ditemukan saat mengirim lalu lintas HTTP/HTTPS ke titik akhir layanan App Runner ... 285
Koneksi gagal ke Amazon RDS atau layanan hilir ... 286
Ketika tidak ada cukup alamat IP untuk peluncuran atau penskalaan ... 289
Cara memperbarui layanan Anda agar lebih banyak tersedia IPs ... 289
Menghitung IPs yang diperlukan untuk layanan Anda ... 290
Buat subnet baru ... 290
Melampirkan blok CIDR Sekunder ke VPC Anda ... 291
Verifikasi ... 292
Perangkap Umum ... 292
Sumber Daya Tambahan ... 293
Glosarium ... 293
Keamanan ... 294
Perlindungan data ... 295
Enkripsi data ... 296
Privasi antar jaringan ... 297
Manajemen identitas dan akses ... 297
Audiens ... 298
Mengautentikasi dengan identitas ... 299
Mengelola akses menggunakan kebijakan ... 302
Pelari Aplikasi dan IAM ... 305
Contoh kebijakan berbasis identitas ... 314
Menggunakan peran terkait layanan ... 319
AWS kebijakan terkelola ... 326
Pemecahan Masalah ... 328
Pencatatan dan pemantauan ... 330
Validasi kepatuhan ... 331
Ketahanan ... 332

Keamanan infrastruktur 332
Titik akhir VPC 333
Menyiapkan titik akhir VPC untuk App Runner 334
Pertimbangan privasi jaringan VPC 334
Menggunakan kebijakan titik akhir untuk mengontrol akses dengan VPC endpoint 335
Mengintegrasikan dengan titik akhir antarmuka 335
Model tanggung jawab bersama 335
Gambar wadah tambalan 335
Praktik terbaik keamanan 336
Praktik terbaik keamanan pencegahan 336
Praktik terbaik keamanan detective 336
AWS Glosarium 338
..... cccxxxix

Apa itu AWS App Runner?

AWS App Runner adalah AWS layanan yang menyediakan cara cepat, sederhana, dan hemat biaya untuk menyebarkan dari kode sumber atau gambar kontainer langsung ke aplikasi web yang dapat diskalakan dan aman di Cloud. AWS Anda tidak perlu mempelajari teknologi baru, memutuskan layanan komputasi mana yang akan digunakan, atau mengetahui cara menyediakan dan mengonfigurasi AWS sumber daya.

App Runner terhubung langsung ke kode atau repositori gambar Anda. Ini menyediakan integrasi otomatis dan pipa pengiriman dengan operasi yang dikelola sepenuhnya, kinerja tinggi, skalabilitas, dan keamanan.



Untuk siapa App Runner?

Jika Anda seorang pengembang, Anda dapat menggunakan App Runner untuk menyederhanakan proses penerapan versi baru kode atau repositori gambar Anda.

Untuk tim operasi, App Runner mengaktifkan penerapan otomatis setiap kali komit didorong ke repositori kode atau versi gambar kontainer baru didorong ke repositori gambar.

Mengakses Pelari Aplikasi

Anda dapat menentukan dan mengonfigurasi penerapan layanan App Runner menggunakan salah satu antarmuka berikut:

- App Runner console — Menyediakan antarmuka web untuk mengelola layanan App Runner Anda.
- App Runner API — Menyediakan RESTful API untuk melakukan tindakan App Runner. Untuk informasi lebih lanjut, lihat Referensi API AWS App Runner.
- AWS Command Line Interface (AWS CLI) - Menyediakan perintah untuk serangkaian AWS layanan yang luas, termasuk Amazon VPC, dan didukung di Windows, macOS, dan Linux. Untuk informasi selengkapnya, lihat AWS Command Line Interface.
- AWS SDKsMenyediakan bahasa khusus APIs dan menangani banyak detail koneksi, seperti menghitung tanda tangan, menangani percobaan ulang permintaan, dan penanganan kesalahan. Untuk informasi selengkapnya, lihat AWS SDKs.

Harga untuk App Runner

App Runner menyediakan cara hemat biaya untuk menjalankan aplikasi Anda. Anda hanya membayar sumber daya yang digunakan oleh layanan App Runner Anda. Layanan Anda menurunkan skala ke instans komputasi yang lebih sedikit saat lalu lintas permintaan lebih rendah. Anda memiliki kontrol atas setelan skalabilitas: jumlah instans yang disediakan terendah dan tertinggi, dan beban tertinggi yang ditangani instans.

Untuk informasi selengkapnya tentang penskalaan otomatis App Runner, lihat. the section called “Penskalaan otomatis”

Untuk informasi harga, lihat Harga AWS App Runner.

Apa selanjutnya

Pelajari cara memulai dengan App Runner dalam topik berikut:

- Pengaturan— Selesaikan langkah-langkah prasyarat untuk menggunakan App Runner.
- Memulai— Terapkan aplikasi pertama Anda ke App Runner.

Menyiapkan untuk App Runner

Jika Anda adalah AWS pelanggan baru, selesaikan prasyarat penyiapan yang tercantum di halaman ini sebelum Anda mulai menggunakannya. AWS App Runner

Untuk prosedur penyiapan ini, Anda menggunakan layanan AWS Identity and Access Management (IAM). Untuk informasi selengkapnya tentang IAM, lihat bahan referensi berikut:

- AWS Identity and Access Management (IAM)
- Panduan Pengguna IAM

Mendaftar untuk Akun AWS

Jika Anda tidak memiliki Akun AWS, selesaikan langkah-langkah berikut untuk membuatnya.

Untuk mendaftar untuk Akun AWS

1. Buka https://portal.aws.amazon.com/billing/pendaftaran.
2. Ikuti petunjuk online.

   Bagian dari prosedur pendaftaran melibatkan menerima panggilan telepon atau pesan teks dan memasukkan kode verifikasi pada keypad telepon.

   Saat Anda mendaftar untuk sebuah Akun AWS, sebuah Pengguna root akun AWSdibuat. Pengguna root memiliki akses ke semua Layanan AWS dan sumber daya di akun. Sebagai praktik keamanan terbaik, tetapkan akses administratif ke pengguna, dan gunakan hanya pengguna root untuk melakukan tugas yang memerlukan akses pengguna root.

AWS mengirimi Anda email konfirmasi setelah proses pendaftaran selesai. Kapan saja, Anda dapat melihat aktivitas akun Anda saat ini dan mengelola akun Anda dengan masuk https://aws.amazon.comke/ dan memilih Akun Saya.

Buat pengguna dengan akses administratif

Setelah Anda mendaftar Akun AWS, amankan Pengguna root akun AWS, aktifkan AWS IAM Identity Center, dan buat pengguna administratif sehingga Anda tidak menggunakan pengguna root untuk tugas sehari-hari.

Amankan Anda Pengguna root akun AWS

1. Masuk ke AWS Management Consolesebagai pemilik akun dengan memilih pengguna Root dan memasukkan alamat Akun AWS email Anda. Di laman berikutnya, masukkan kata sandi.

   Untuk bantuan masuk dengan menggunakan pengguna root, lihat Masuk sebagai pengguna root di AWS Sign-In Panduan Pengguna.

2. Mengaktifkan autentikasi multi-faktor (MFA) untuk pengguna root Anda.

   Untuk petunjuk, lihat Mengaktifkan perangkat MFA virtual untuk pengguna Akun AWS root (konsol) Anda di Panduan Pengguna IAM.

Buat pengguna dengan akses administratif

1. Aktifkan Pusat Identitas IAM.

   Untuk mendapatkan petunjuk, silakan lihat Mengaktifkan AWS IAM Identity Center di Panduan Pengguna AWS IAM Identity Center .

2. Di Pusat Identitas IAM, berikan akses administratif ke pengguna.

   Untuk tutorial tentang menggunakan Direktori Pusat Identitas IAM sebagai sumber identitas Anda, lihat Mengkonfigurasi akses pengguna dengan default Direktori Pusat Identitas IAM di Panduan AWS IAM Identity Center Pengguna.

Masuk sebagai pengguna dengan akses administratif

- Untuk masuk dengan pengguna Pusat Identitas IAM, gunakan URL masuk yang dikirim ke alamat email saat Anda membuat pengguna Pusat Identitas IAM.

  Untuk bantuan masuk menggunakan pengguna Pusat Identitas IAM, lihat Masuk ke portal AWS akses di Panduan AWS Sign-In Pengguna.

Tetapkan akses ke pengguna tambahan

1. Di Pusat Identitas IAM, buat set izin yang mengikuti praktik terbaik menerapkan izin hak istimewa paling sedikit.

   Untuk petunjuknya, lihat Membuat set izin di Panduan AWS IAM Identity Center Pengguna.

2. Tetapkan pengguna ke grup, lalu tetapkan akses masuk tunggal ke grup.

   Untuk petunjuk, lihat Menambahkan grup di Panduan AWS IAM Identity Center Pengguna.

Memberikan akses programatis

Pengguna membutuhkan akses terprogram jika mereka ingin berinteraksi dengan AWS luar. AWS Management Console Cara untuk memberikan akses terprogram tergantung pada jenis pengguna yang mengakses AWS.

Untuk memberi pengguna akses programatis, pilih salah satu opsi berikut.

Pengguna mana yang membutuhkan akses programatis?	Untuk	Oleh
Identitas tenaga kerja (Pengguna yang dikelola di Pusat Identitas IAM)	Gunakan kredensi sementara untuk menandatangani permintaan terprogram ke AWS CLI,, AWS SDKs atau. AWS APIs	Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan. • Untuk AWS CLI, lihat Mengkonfigurasi yang akan AWS CLI digunakan AWS IAM Identity Center dalam Panduan AWS Command Line Interface Pengguna. • Untuk AWS SDKs, alat, dan AWS APIs, lihat Autentika si Pusat Identitas IAM di Panduan Referensi Alat AWS SDKs dan Alat.
IAM	Gunakan kredensi sementara untuk menandatangani permintaan terprogram ke AWS CLI,, AWS SDKs atau. AWS APIs	Mengikuti petunjuk dalam Menggunakan kredensil sementara dengan AWS sumber daya di Panduan Pengguna IAM.

Pengguna mana yang membutuhkan akses programatis?	Untuk	Oleh
IAM	(Tidak direkomendasikan) Gunakan kredensi jangka panjang untuk menandata ngani permintaan terprogra m ke AWS CLI,, AWS SDKs atau. AWS APIs	Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan. • Untuk mengetahui AWS CLI, lihat Mengautentikasi menggunakan kredensil pengguna IAM di Panduan Pengguna.AWS Command Line Interface • Untuk AWS SDKs dan alat, lihat Mengautentikasi menggunakan kredensil jangka panjang di Panduan Referensi Alat AWS SDKs dan Alat. • Untuk AWS APIs, lihat Mengelola kunci akses untuk pengguna IAM di Panduan Pengguna IAM.

Apa selanjutnya

Anda menyelesaikan langkah-langkah prasyarat. Untuk menerapkan aplikasi pertama Anda ke App Runner, lihat. Memulai

Memulai dengan App Runner

AWS App Runner adalah AWS layanan yang menyediakan cara cepat, sederhana, dan hemat biaya untuk mengubah gambar kontainer atau kode sumber yang ada langsung menjadi layanan web yang berjalan di situs web. AWS Cloud

Tutorial ini mencakup bagaimana Anda dapat menggunakan AWS App Runner untuk menyebarkan aplikasi Anda ke layanan App Runner. Ini berjalan melalui konfigurasi kode sumber dan penerapan, build layanan, dan runtime layanan. Ini juga menunjukkan cara menerapkan versi kode, membuat perubahan konfigurasi, dan melihat log. Terakhir, tutorial menunjukkan cara membersihkan sumber daya yang Anda buat saat mengikuti prosedur tutorial.

Topik

- Prasyarat
- Langkah 1: Buat layanan App Runner
- Langkah 2: Ubah kode layanan Anda
- Langkah 3: Buat perubahan konfigurasi
- Langkah 4: Lihat log untuk layanan Anda
- Langkah 5: Bersihkan
- Apa selanjutnya

Prasyarat

Sebelum Anda memulai tutorial, pastikan untuk mengambil tindakan berikut:

1. Selesaikan langkah-langkah pengaturan diPengaturan.
2. Putuskan apakah Anda ingin bekerja dengan repositori atau GitHub repositori Bitbucket.
   - Untuk bekerja dengan Bitbucket, pertama-tama buat akun Bitbucket, jika Anda belum memilikinya. Jika Anda baru mengenal Bitbucket, lihat Memulai Bitbucket di Dokumentasi Cloud Bitbucket.
   - Untuk bekerja dengan GitHub, buat GitHubakun, jika Anda belum memilikinya. Jika Anda baru mengenal GitHub, lihat Memulai GitHub di GitHubDokumen.

> **Note**
>
> Anda dapat membuat koneksi ke beberapa penyedia repositori dari akun Anda. Jadi, jika Anda ingin berjalan melalui penerapan dari repositori GitHub dan Bitbucket, Anda dapat mengulangi prosedur ini. Lain kali melalui membuat layanan App Runner baru dan membuat koneksi akun baru untuk penyedia repositori lainnya.

3. Buat repositori di akun penyedia repositori Anda. Tutorial ini menggunakan nama repositori. `python-hello` Buat file di direktori root repositori, dengan nama dan konten yang ditentukan dalam contoh berikut.

File untuk **`python-hello`** repositori contoh

Example requirements.txt

```
pyramid==2.0
```

Example server.py

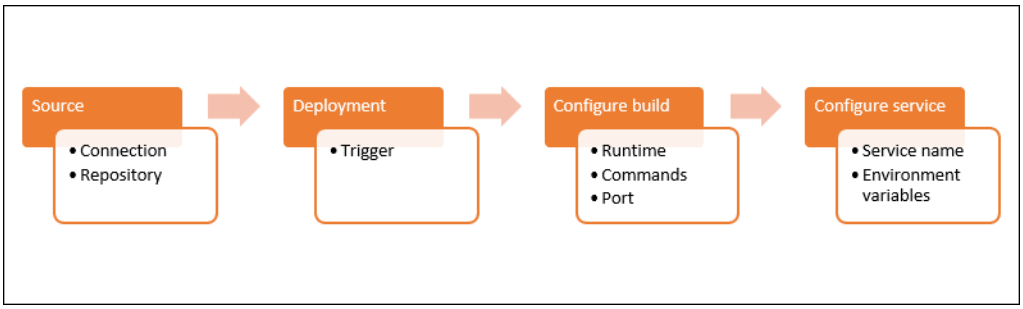
```
from wsgiref.simple_server import make_server
from pyramid.config import Configurator
from pyramid.response import Response
import os

def hello_world(request):
    name = os.environ.get('NAME')
    if name == None or len(name) == 0:
        name = "world"
    message = "Hello, " + name + "!\n"
    return Response(message)

if __name__ == '__main__':
    port = int(os.environ.get("PORT"))
    with Configurator() as config:
        config.add_route('hello', '/')
        config.add_view(hello_world, route_name='hello')
        app = config.make_wsgi_app()
    server = make_server('0.0.0.0', port, app)
    server.serve_forever()
```

Langkah 1: Buat layanan App Runner

Pada langkah ini, Anda membuat layanan App Runner berdasarkan contoh repositori kode sumber yang Anda buat GitHub atau Bitbucket sebagai bagian dari. the section called “Prasyarat” Contohnya berisi situs web Python sederhana. Ini adalah langkah-langkah utama yang Anda ambil untuk membuat layanan:

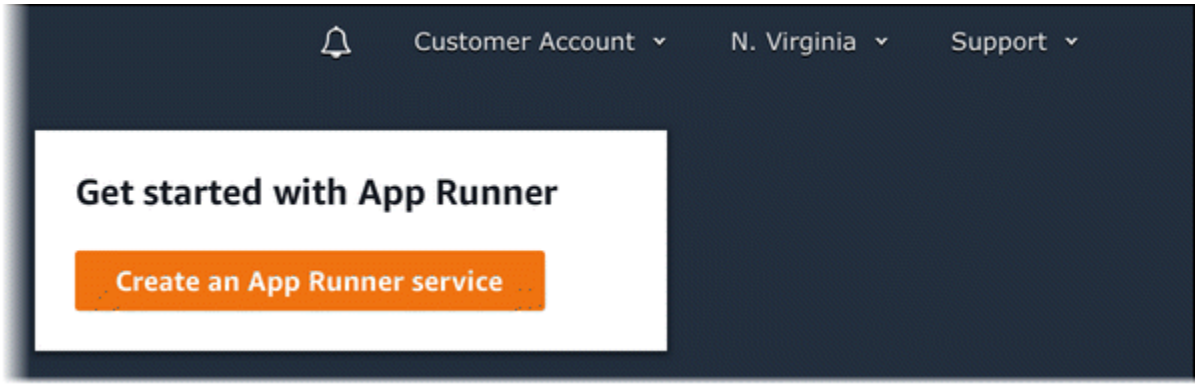
1. Konfigurasikan kode sumber Anda.
2. Konfigurasikan penyebaran sumber.
3. Konfigurasikan pembuatan aplikasi.
4. Konfigurasikan layanan Anda.
5. Tinjau dan konfirmasi.

Diagram berikut menguraikan langkah-langkah untuk membuat layanan App Runner:

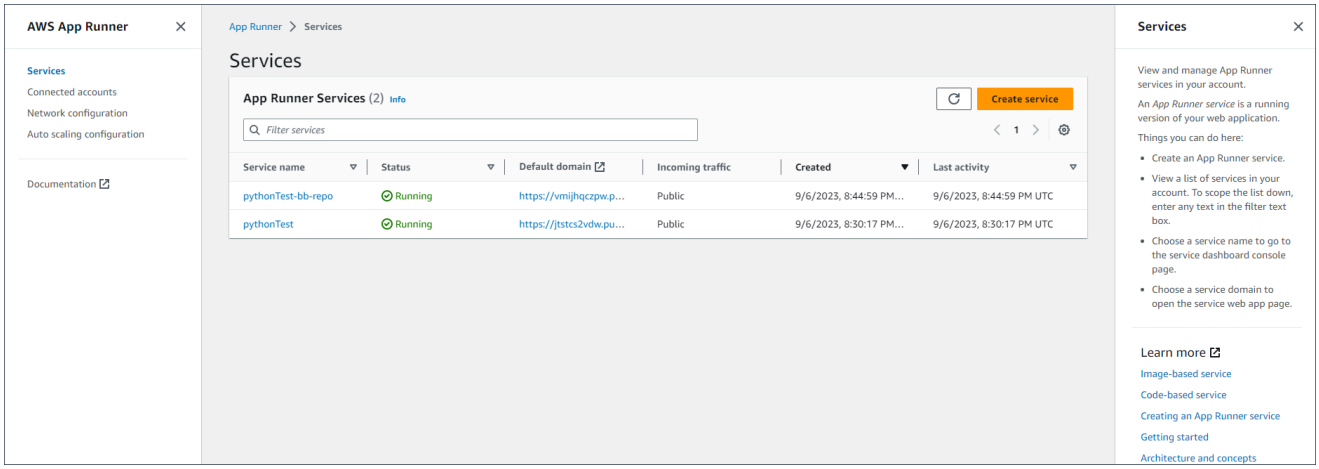


Untuk membuat layanan App Runner berdasarkan repositori kode sumber

1. Konfigurasikan kode sumber Anda.
   a. Buka konsol App Runner, dan di daftar Wilayah, pilih Wilayah AWS.
   b. Jika Akun AWS belum memiliki layanan App Runner, halaman beranda konsol akan ditampilkan. Pilih Buat layanan App Runner.



Jika Akun AWS memiliki layanan yang ada, halaman Layanan dengan daftar layanan Anda akan ditampilkan. Pilih Buat layanan.



c. Pada halaman Sumber dan penyebaran, di bagian Sumber, untuk jenis Repositori, pilih repositori kode sumber.

d. Pilih Jenis Penyedia. Pilih salah satu GitHubatau Bitbucket.

e. Selanjutnya pilih Tambah baru. Jika diminta, berikan kredensi Anda GitHub atau Bitbucket.

f. Pilih rangkaian langkah berikutnya berdasarkan jenis Penyedia yang Anda pilih sebelumnya.

> **Note**
>
> Langkah-langkah berikut untuk menginstal konektor AWS GitHub ke GitHub akun Anda adalah langkah satu kali. Anda dapat menggunakan kembali koneksi untuk membuat beberapa layanan App Runner berdasarkan repositori di akun ini. Ketika Anda memiliki koneksi yang ada, pilih dan lewati ke pemilihan repositori.
> Hal yang sama berlaku untuk konektor AWS untuk akun Bitbucket Anda. Jika Anda menggunakan keduanya GitHub dan Bitbucket sebagai repositori kode sumber untuk layanan App Runner, Anda harus menginstal satu AWS Connector untuk

setiap penyedia. Anda kemudian dapat menggunakan kembali setiap konektor untuk membuat lebih banyak layanan App Runner.

- Untuk GitHub, ikuti langkah-langkah ini.
  i. Pada layar berikutnya, masukkan Nama Koneksi.
  ii. Jika ini pertama kali Anda gunakan GitHub dengan App Runner, pilih Instal yang lain.
  iii. Di kotak GitHub dialog AWS Konektor untuk, jika diminta, pilih nama GitHub akun Anda.
  iv. Jika diminta untuk mengotorisasi AWS Konektor GitHub, pilih Otorisasi Koneksi AWS.
  v. Dalam Instal AWS Konektor untuk kotak GitHub dialog, Pilih Instal.

     Nama akun Anda muncul sebagai GitHub akun/organisasi yang dipilih. Anda sekarang dapat memilih repositori di akun Anda.
  vi. Untuk Repositori, pilih contoh repositori yang Anda buat,. `python-hello` Untuk Branch, pilih nama cabang default repositori Anda (misalnya, main).
  vii. Tinggalkan direktori Source dengan nilai default. Direktori default ke root repositori. Anda menyimpan kode sumber Anda di direktori root repositori di langkah-langkah Prasyarat sebelumnya.
- Untuk Bitbucket, ikuti langkah-langkah ini.
  i. Pada layar berikutnya, masukkan Nama Koneksi.
  ii. Jika ini pertama kalinya Anda menggunakan Bitbucket dengan App Runner, pilih Instal yang lain.
  iii. Di kotak dialog AWS CodeStar request access, Anda dapat memilih ruang kerja dan memberikan akses ke integrasi AWS CodeStar Bitbucket. Pilih ruang kerja Anda, lalu pilih Grant access.
  iv. Selanjutnya Anda akan diarahkan ke AWS konsol. Verifikasi bahwa aplikasi Bitbucket diatur ke ruang kerja Bitbucket yang benar dan pilih Berikutnya.
  v. Untuk Repositori, pilih contoh repositori yang Anda buat,. `python-hello` Untuk Branch, pilih nama cabang default repositori Anda (misalnya, main).

vi. Tinggalkan direktori Source dengan nilai default. Direktori default ke root repositori. Anda menyimpan kode sumber Anda di direktori root repositori di langkah-langkah Prasyarat sebelumnya.

2. Konfigurasikan penerapan Anda: Di bagian Pengaturan Deployment, pilih Otomatis, lalu pilih Berikutnya.

> **Note**
>
> Dengan penerapan otomatis, setiap komit baru ke direktori sumber repositori Anda secara otomatis menerapkan versi baru layanan Anda.

Source and deployment Info

Choose the source for your App Runner service and the way it's deployed.

Source and deployment

Source

Repository type

- ○ Container registry
  Deploy your service using a container image stored in a container registry.
- ◉ Source code repository
  Deploy your service using the code hosted in a source repository.

Provider
Choose the provider where you host your code repository.

GitHub

Github Connection Info

App Runner deploys your source code by installing an app called "AWS Connector for GitHub" in your account. You can install this app in your main GitHub account or in a GitHub organization.

myGitHub [Add new]

Repository

python-hello

Branch

main

Source directory
The build and start commands will execute in this directory. App Runner defaults to the root directory if you don't specify a directory here.

/

Leading and trailing slashes ("/") are not required. Valid examples: "apps/targetapp", "/apps/targetapp/", "/targetapp"

Deployment settings

Deployment trigger

- ○ Manual
  Start each deployment yourself using the App Runner console or AWS CLI.
- ◉ Automatic
  Every push to this branch that affects files in the specified **Source directory** deploys a new version of your service.

Cancel Next

3. Konfigurasikan pembuatan aplikasi.

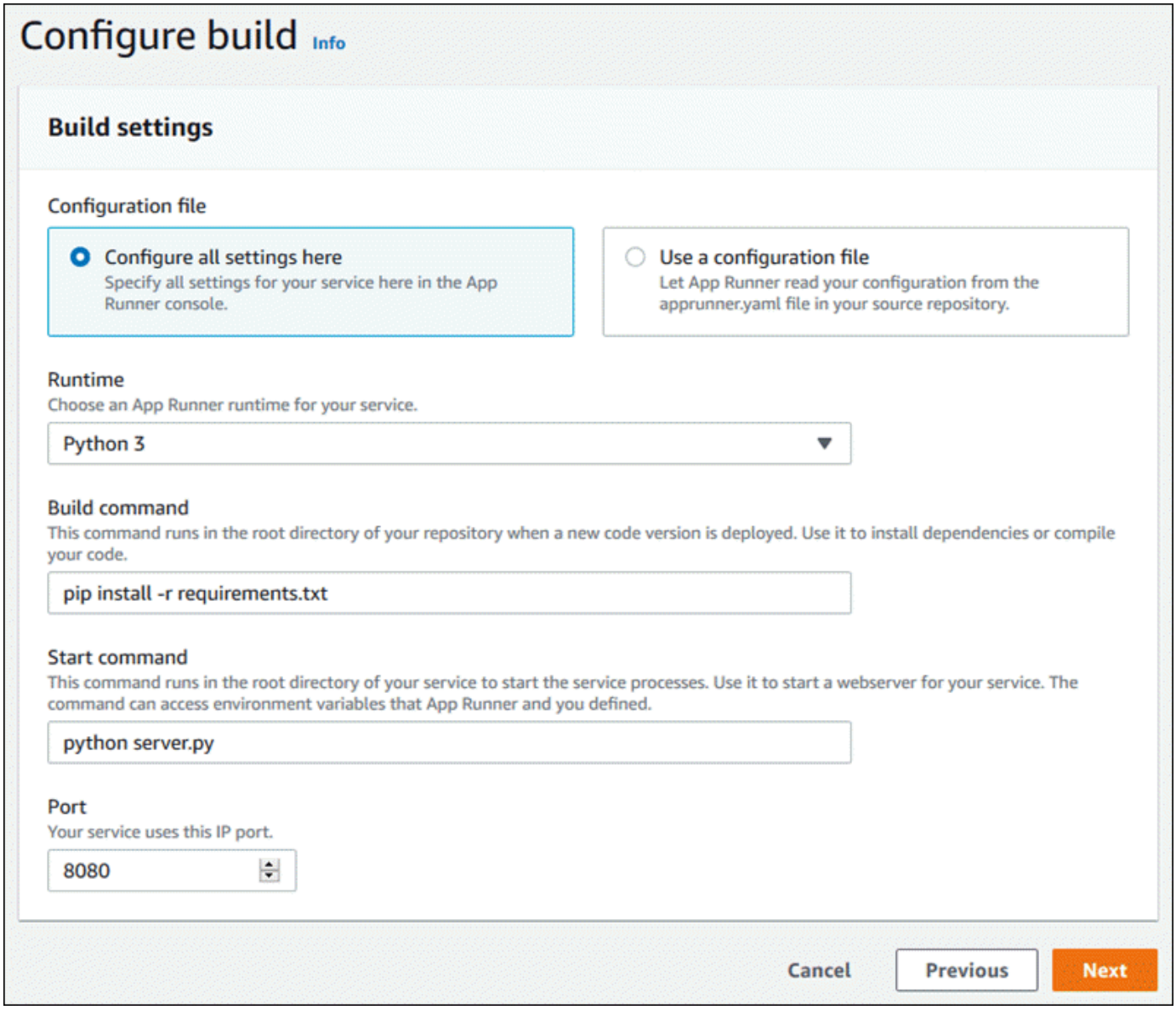
   a. Pada halaman Konfigurasi build, untuk file Konfigurasi, pilih Konfigurasi semua pengaturan di sini.

   b. Berikan setelan build berikut:

      - Runtime - Pilih Python 3.
      - Membangun perintah — Enter**`pip install -r requirements.txt`**.
      - Mulai perintah — Enter**`python server.py`**.
      - Pelabuhan — Masuk**`8080`**.

   c. Pilih Berikutnya.

> **Note**
>
> Runtime Python 3 membangun image Docker menggunakan gambar dasar Python 3 dan contoh kode Python Anda. Kemudian meluncurkan layanan yang menjalankan instance container dari gambar ini.



4. Konfigurasikan layanan Anda.

   a. Pada halaman Konfigurasi Layanan, di bagian Pengaturan layanan, masukkan nama layanan.

   b. Di bawah Variabel lingkungan, pilih Tambahkan variabel lingkungan. Berikan nilai-nilai berikut untuk variabel lingkungan.

      - Sumber - Pilih Teks biasa
      - Nama variabel lingkungan - **`NAME`**
      - Nilai variabel lingkungan - nama apa pun (misalnya, nama depan Anda).

> **Note**
>
> Contoh aplikasi membaca nama yang Anda tetapkan dalam variabel lingkungan ini dan menampilkan nama di halaman webnya.

c. Pilih Berikutnya.

Configure service Info

Service settings

Service name

python-test

Virtual CPU & memory

1 vCPU | 2 GB

Environment variables — *optional* Info

Add environment variables in plain text or reference them from Secrets Manager and SSM Parameter Store. Update IAM Policies using the IAM Policy template given below to securely reference secrets and configurations as environment variables.

No environment variables have been configured.

Add environment variable

You can add up to 50 more items.

▶ IAM policy templates

▶ Auto scaling Info

Configure automatic scaling behavior.

▶ Health check Info

Configure load balancer health checks.

▶ Security Info

Specify an Instance role and an AWS KMS encryption key

▶ Networking Info

Configure the way your service communicates with other applications, services, and resources.

▶ Observability

Configure observability tooling.

▼ Tags Info

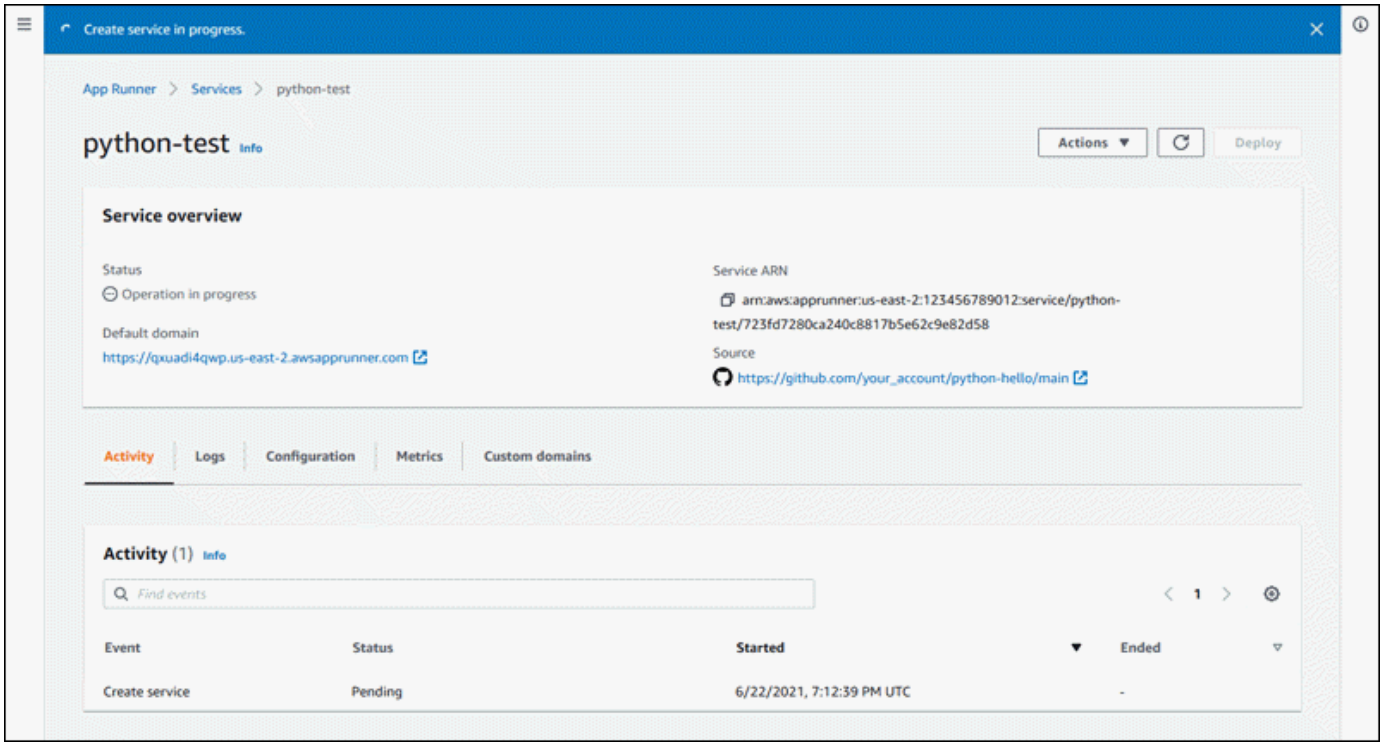
Use tags to search and filter your resources, track your AWS costs, and control access permissions.

Tags — *optional*

A tag is a key-value pair that you assign to an AWS resource.

5. Pada halaman Tinjau dan buat, verifikasi semua detail yang Anda masukkan, lalu pilih Buat dan terapkan.

   Jika layanan berhasil dibuat, konsol menampilkan dasbor layanan, dengan ikhtisar Layanan dari layanan baru.



6. Verifikasi bahwa layanan Anda sedang berjalan.

   a. Pada halaman dasbor layanan, tunggu hingga Status layanan berjalan.

   b. Pilih nilai domain default — ini adalah URL ke situs web layanan Anda.

> **Note**
>
> Untuk meningkatkan keamanan aplikasi App Runner Anda, domain*.awsapprunner.com terdaftar di Daftar Akhiran Publik (PSL). Untuk keamanan lebih lanjut, kami menyarankan Anda menggunakan cookie dengan `__Host-` awalan jika Anda perlu mengatur cookie sensitif di nama domain default untuk aplikasi App Runner Anda. Praktik ini akan membantu mempertahankan domain Anda dari upaya pemalsuan permintaan lintas situs (CSRF). Untuk informasi selengkapnya, lihat halaman Set-Cookie di Jaringan Pengembang Mozilla.

Sebuah halaman web menampilkan: Halo,! ***your name***

Langkah 2: Ubah kode layanan Anda

Pada langkah ini, Anda membuat perubahan pada kode Anda di direktori sumber repositori. Kemampuan CI/CD App Runner secara otomatis membangun dan menyebarkan perubahan ke layanan Anda.

Untuk membuat perubahan pada kode layanan Anda

1. Arahkan ke repositori contoh Anda.
2. Edit file bernama`server.py`.
3. Dalam ekspresi yang ditugaskan ke variabel`message`, ubah teks `Hello` menjadi`Good morning`.
4. Simpan dan komit perubahan Anda ke repositori.
5. Langkah-langkah berikut menggambarkan mengubah kode layanan dalam GitHub repositori.
   a. Arahkan ke GitHub repositori contoh Anda.
   b. Pilih nama file `server.py` untuk menavigasi ke file itu.
   c. Pilih Edit file ini (ikon pensil).
   d. Dalam ekspresi yang ditugaskan ke variabel`message`, ubah teks `Hello` menjadi`Good morning`.

e. Pilih Perubahan commit.

6. Komit baru mulai diterapkan untuk layanan App Runner Anda. Pada halaman dasbor layanan, Status layanan berubah menjadi Operasi yang sedang berlangsung.

   Tunggu hingga penerapan berakhir. Pada halaman dasbor layanan, Status layanan harus berubah kembali ke Running.

7. Verifikasi bahwa penerapan berhasil: segarkan tab browser tempat halaman web layanan Anda ditampilkan.

   Halaman sekarang menampilkan pesan yang dimodifikasi: Selamat pagi,***your name***!

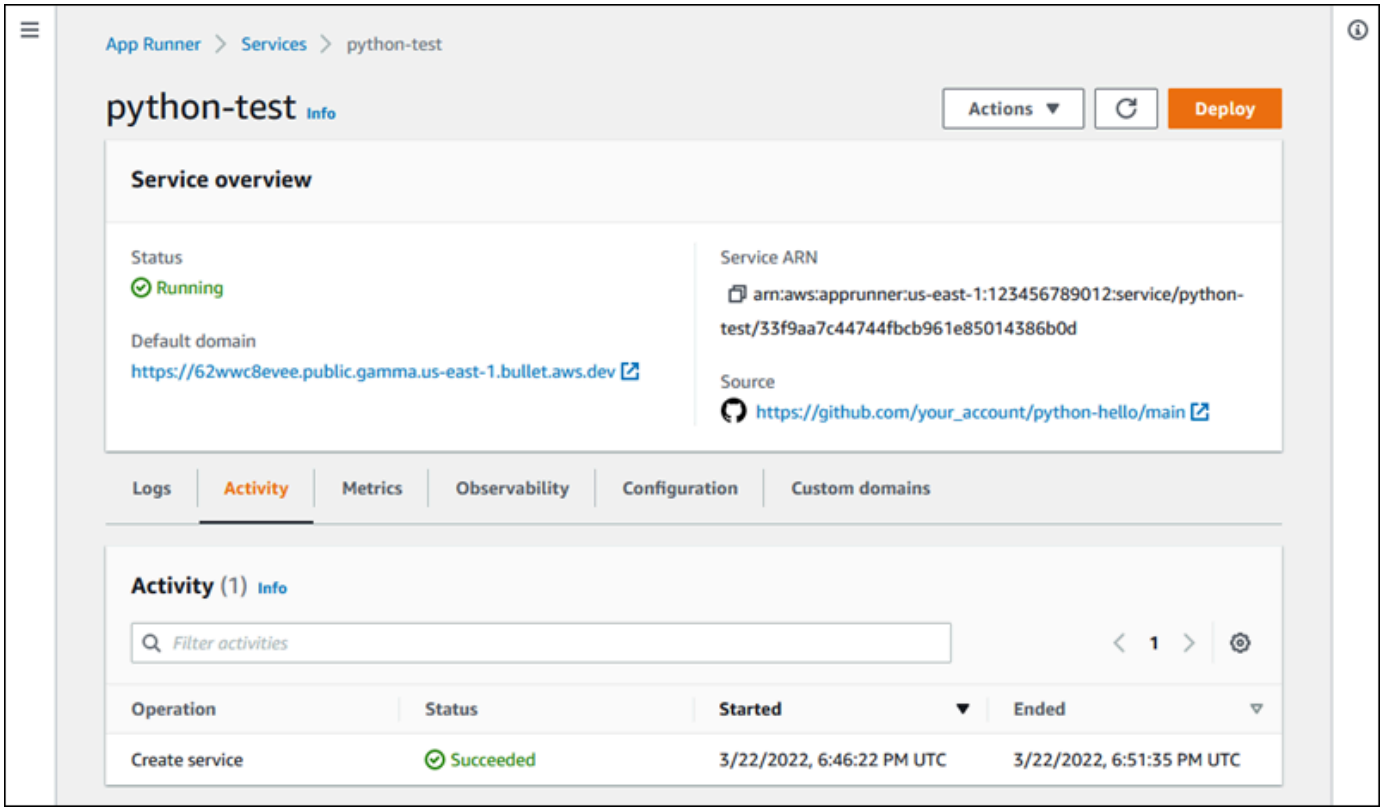
Langkah 3: Buat perubahan konfigurasi

Pada langkah ini, Anda membuat perubahan pada nilai variabel **NAME** lingkungan, untuk menunjukkan perubahan konfigurasi layanan.

Untuk mengubah nilai variabel lingkungan

1. Buka konsol App Runner, dan di daftar Wilayah, pilih Wilayah AWS.
2. Di panel navigasi, pilih Layanan, lalu pilih layanan Pelari Aplikasi Anda.

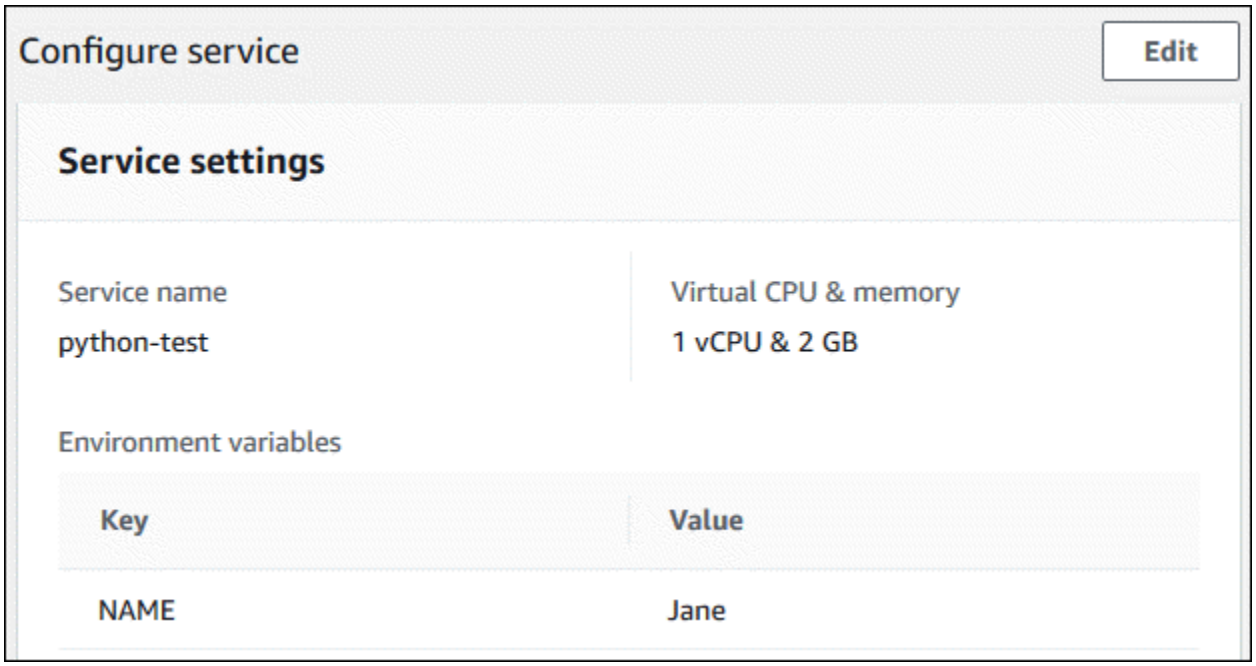
   Konsol menampilkan dasbor layanan dengan ikhtisar Layanan.



3. Pada halaman dasbor layanan, pilih tab Konfigurasi.

   Konsol menampilkan pengaturan konfigurasi layanan Anda di beberapa bagian.

4. Di bagian Konfigurasi layanan, pilih Edit.



5. Untuk variabel lingkungan dengan kunci**`NAME`**, ubah nilai ke nama yang berbeda.

6. Pilih Terapkan perubahan.

   App Runner memulai proses pembaruan. Pada halaman dasbor layanan, Status layanan berubah menjadi Operasi yang sedang berlangsung.

7. Tunggu hingga pembaruan berakhir. Pada halaman dasbor layanan, Status layanan harus berubah kembali ke Running.

8. Verifikasi bahwa pembaruan berhasil: segarkan tab browser tempat halaman web layanan Anda ditampilkan.

   Halaman sekarang menampilkan nama yang dimodifikasi: Selamat pagi,***new name***!

Langkah 4: Lihat log untuk layanan Anda

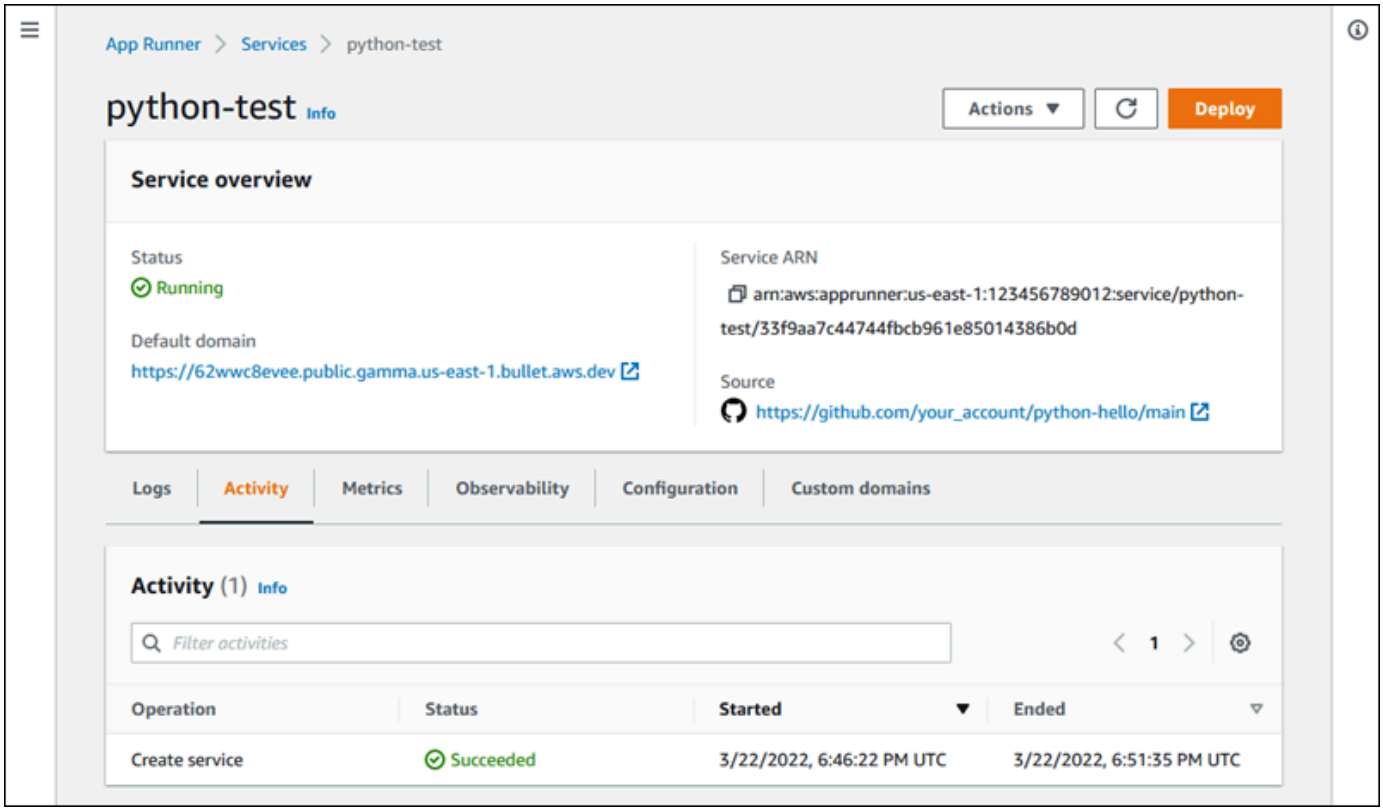
Pada langkah ini, Anda menggunakan konsol App Runner untuk melihat log untuk layanan App Runner Anda. App Runner mengalirkan log ke Amazon CloudWatch Logs (CloudWatch Log) dan menampilkannya di dasbor layanan Anda. Untuk informasi tentang log Pelari Aplikasi, lihatthe section called “Log (CloudWatch Log)”.

Untuk melihat log untuk layanan Anda

1. Buka konsol App Runner, dan di daftar Wilayah, pilih Wilayah AWS.

2. Di panel navigasi, pilih Layanan, lalu pilih layanan Pelari Aplikasi Anda.

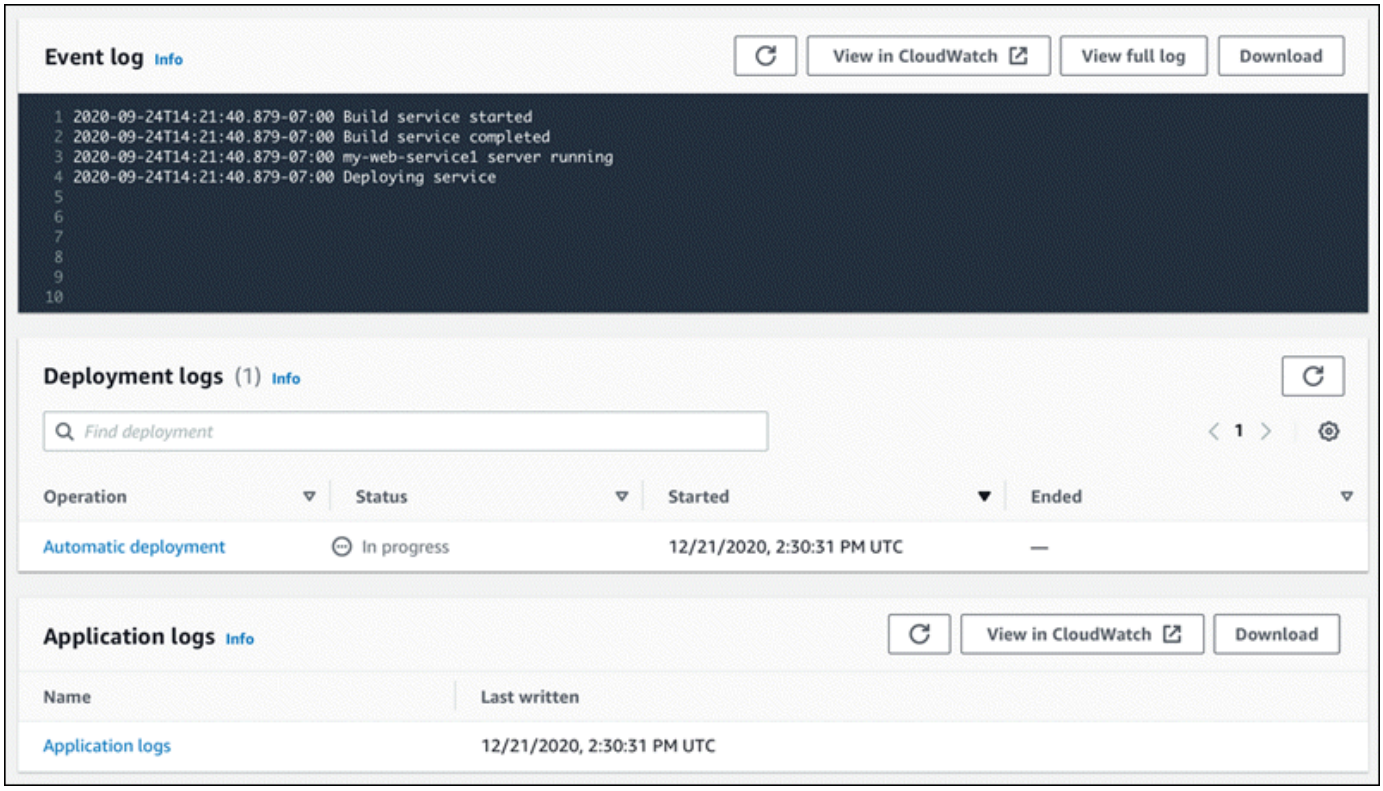
   Konsol menampilkan dasbor layanan dengan ikhtisar Layanan.



3. Pada halaman dasbor layanan, pilih tab Log.

   Konsol menampilkan beberapa jenis log di beberapa bagian:

   - Log peristiwa — Aktivitas dalam siklus hidup layanan App Runner Anda. Konsol menampilkan acara terbaru.
   - Log penerapan — Penerapan repositori sumber ke layanan App Runner Anda. Konsol menampilkan aliran log terpisah untuk setiap penerapan.
   - Log aplikasi — Output dari aplikasi web yang diterapkan ke layanan App Runner Anda. Konsol menggabungkan output dari semua instance yang berjalan ke dalam satu aliran log.



4. Untuk menemukan penerapan tertentu, cakupkan daftar log penerapan dengan memasukkan istilah penelusuran. Anda dapat mencari nilai apa pun yang muncul di tabel.
5. Untuk melihat konten log, pilih Lihat log lengkap (log peristiwa) atau nama aliran log (penerapan dan log aplikasi).
6. Pilih Unduh untuk mengunduh log. Untuk aliran log penerapan, pilih aliran log terlebih dahulu.
7. Pilih Lihat CloudWatch untuk membuka CloudWatch konsol dan gunakan kemampuan penuhnya untuk menjelajahi log layanan App Runner Anda. Untuk aliran log penerapan, pilih aliran log terlebih dahulu.

> **Note**
>
> CloudWatch Konsol ini sangat berguna jika Anda ingin melihat log aplikasi dari instance tertentu alih-alih log aplikasi gabungan.

Langkah 5: Bersihkan

Anda sekarang telah mempelajari cara membuat layanan App Runner, melihat log, dan membuat beberapa perubahan. Pada langkah ini, Anda menghapus layanan untuk menghapus sumber daya yang tidak Anda butuhkan lagi.

Untuk menghapus layanan Anda

1. Pada halaman dasbor layanan, pilih Tindakan, lalu pilih Hapus layanan.
2. Dalam dialog konfirmasi, masukkan teks yang diminta, lalu pilih Hapus.

   Hasil: Konsol menavigasi ke halaman Layanan. Layanan yang baru saja Anda hapus menunjukkan status MENGHAPUS. Beberapa saat kemudian menghilang dari daftar.

Pertimbangkan juga menghapus koneksi GitHub dan Bitbucket yang Anda buat sebagai bagian dari tutorial ini. Untuk informasi selengkapnya, lihat the section called “Koneksi”.

Apa selanjutnya

Setelah menerapkan layanan App Runner pertama Anda, pelajari lebih lanjut dalam topik berikut:

- Arsitektur dan konsepArsitektur, konsep utama, dan AWS sumber daya yang terkait dengan App Runner.
- Layanan berbasis gambardan Layanan berbasis kode — Dua jenis sumber aplikasi yang dapat diterapkan oleh App Runner.
- Mengembangkan untuk App Runner— Hal-hal yang harus Anda ketahui saat mengembangkan atau memigrasi kode aplikasi untuk penerapan ke App Runner.
- Konsol Pelari Aplikasi— Kelola dan pantau layanan Anda menggunakan konsol App Runner.
- Mengelola layanan Anda— Kelola siklus hidup layanan App Runner Anda.
- Observabilitas— Dapatkan visibilitas ke dalam operasi layanan App Runner Anda dengan memantau metrik, membaca log, menangani peristiwa, melacak panggilan tindakan layanan, dan melacak peristiwa aplikasi seperti panggilan HTTP.
- File konfigurasi App Runner— Cara berbasis konfigurasi untuk menentukan opsi untuk perilaku build dan runtime layanan App Runner Anda.
- API Pelari Aplikasi— Gunakan antarmuka pemrograman aplikasi (API) App Runner untuk membuat, membaca, memperbarui, dan menghapus sumber daya App Runner.

- Keamanan— Berbagai cara itu AWS dan Anda memastikan keamanan cloud saat Anda menggunakan App Runner dan layanan lainnya.

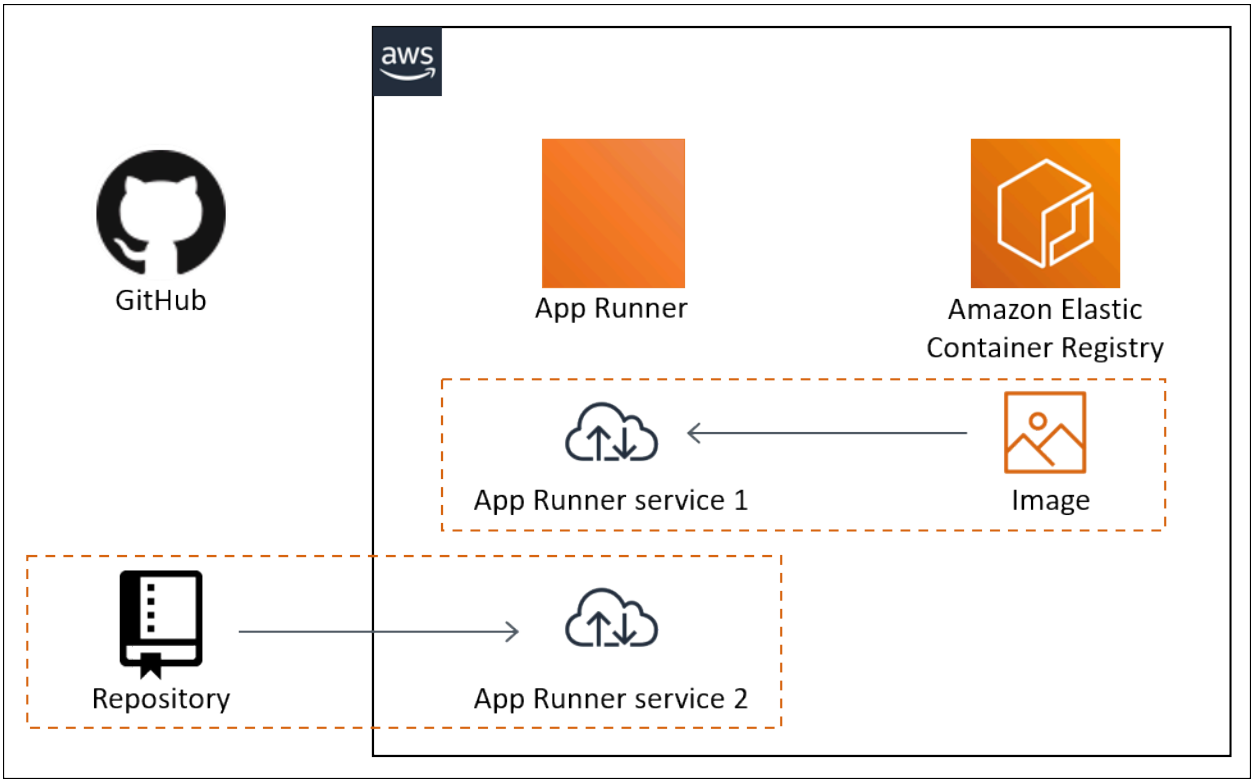
Arsitektur dan konsep App Runner

AWS App Runner mengambil kode sumber atau gambar sumber Anda dari repositori, dan membuat dan memelihara layanan web yang berjalan untuk Anda di. AWS Cloud Biasanya, Anda perlu memanggil hanya satu tindakan App Runner, CreateService, untuk membuat layanan Anda.

Dengan repositori gambar sumber, Anda menyediakan gambar ready-to-use kontainer yang dapat diterapkan oleh App Runner untuk menjalankan layanan web Anda. Dengan repositori kode sumber, Anda memberikan kode dan instruksi untuk membangun dan menjalankan layanan web, dan Anda menargetkan lingkungan runtime tertentu. App Runner mendukung beberapa platform pemrograman, masing-masing dengan satu atau lebih runtime terkelola untuk versi utama platform.

Pada saat ini, App Runner dapat mengambil kode sumber Anda dari Bitbucket atau GitHubrepositori, atau dapat mengambil gambar sumber Anda dari Amazon Elastic Container Registry (Amazon ECR) di situs Anda. Akun AWS

Diagram berikut menunjukkan ikhtisar arsitektur layanan App Runner. Dalam diagram, ada dua contoh layanan: satu menyebarkan kode sumber dari GitHub, dan yang lainnya menyebarkan gambar sumber dari Amazon ECR. Aliran yang sama berlaku untuk repositori Bitbucket.



Konsep App Runner

Berikut ini adalah konsep kunci yang terkait dengan layanan web Anda yang berjalan di App Runner:

- Layanan Pelari Aplikasi — AWS Sumber daya yang digunakan App Runner untuk menyebarkan dan mengelola aplikasi Anda berdasarkan repositori kode sumber atau gambar kontainer. Layanan App Runner adalah versi aplikasi yang sedang berjalan. Untuk informasi selengkapnya tentang membuat layanan, lihatthe section called “Pembuatan”.
- Jenis sumber — Jenis repositori sumber yang Anda sediakan untuk menerapkan layanan App Runner: kode sumber atau gambar sumber.
- Penyedia repositori — Layanan repositori yang berisi sumber aplikasi Anda (misalnya, Bitbucket GitHub, atau Amazon ECR).
- Koneksi App Runner — AWS Sumber daya yang memungkinkan App Runner mengakses akun penyedia repositori (misalnya, akun atau GitHub organisasi). Untuk informasi selengkapnya tentang koneksi, lihatthe section called “Koneksi”.
- Runtime — Gambar dasar untuk menyebarkan repositori kode sumber. App Runner menyediakan berbagai runtime terkelola untuk berbagai platform dan versi pemrograman. Untuk informasi selengkapnya, lihat Layanan berbasis kode.
- Deployment — Tindakan yang menerapkan versi repositori sumber Anda (kode atau gambar) ke layanan App Runner. Penyebaran pertama ke layanan terjadi sebagai bagian dari pembuatan layanan. Penerapan selanjutnya dapat terjadi dengan salah satu dari dua cara:
  - Penyebaran otomatis - Kemampuan CI/CD. Anda dapat mengonfigurasi layanan App Runner untuk secara otomatis membangun (untuk kode sumber) dan menerapkan setiap versi aplikasi Anda seperti yang muncul di repositori. Ini bisa berupa komit baru di repositori kode sumber atau versi gambar baru di repositori gambar sumber.
  - Penerapan manual — Penerapan ke layanan App Runner yang Anda mulai secara eksplisit.
- Domain kustom — Domain yang Anda kaitkan dengan layanan App Runner Anda. Pengguna aplikasi web Anda dapat menggunakan domain ini untuk mengakses layanan web Anda, bukan subdomain App Runner default. Untuk informasi selengkapnya, lihat the section called “Nama domain khusus”.

> **Note**
>
> Untuk meningkatkan keamanan aplikasi App Runner Anda, domain*.awsapprunner.com terdaftar di Daftar Akhiran Publik (PSL). Untuk keamanan lebih lanjut, kami menyarankan Anda menggunakan cookie dengan `__Host-` awalan jika Anda perlu mengatur cookie

sensitif di nama domain default untuk aplikasi App Runner Anda. Praktik ini akan membantu mempertahankan domain Anda dari upaya pemalsuan permintaan lintas situs (CSRF). Untuk informasi selengkapnya, lihat halaman Set-Cookie di Jaringan Pengembang Mozilla.

- Pemeliharaan — Aktivitas yang sesekali dilakukan oleh App Runner pada infrastruktur yang menjalankan layanan App Runner Anda. Saat pemeliharaan sedang berlangsung, status layanan sementara berubah menjadi `OPERATION_IN_PROGRESS` (Operasi sedang berlangsung di konsol) selama beberapa menit. Tindakan pada layanan Anda (misalnya, penerapan, pembaruan konfigurasi, jeda/lanjutkan, atau penghapusan) diblokir selama waktu ini. Coba tindakan lagi beberapa menit kemudian, ketika status layanan kembali ke`RUNNING`.

  > **Note**
  >
  > Jika tindakan Anda gagal, itu tidak berarti bahwa layanan App Runner Anda sedang down. Aplikasi Anda aktif dan terus menangani permintaan. Tidak mungkin layanan Anda mengalami downtime.

  Secara khusus, App Runner memigrasikan layanan Anda jika mendeteksi masalah pada perangkat keras yang mendasari hosting layanan. Untuk mencegah downtime layanan apa pun, App Runner menerapkan layanan Anda ke kumpulan instance baru dan mengalihkan lalu lintas ke instans (penerapan biru-hijau). Anda mungkin kadang-kadang melihat sedikit peningkatan sementara dalam biaya.

Konfigurasi yang didukung App Runner

Saat mengonfigurasi layanan App Runner, Anda menentukan CPU virtual dan konfigurasi memori yang akan dialokasikan ke layanan Anda. Anda membayar berdasarkan konfigurasi komputasi yang Anda pilih. Untuk informasi lebih lanjut tentang harga, lihat AWS Resource Groups Harga.

Tabel berikut menyediakan informasi tentang vCPU dan konfigurasi memori yang didukung App Runner:

CPU	Memori
0,25 vCPU	0,5 GB

CPU	Memori
0,25 vCPU	1 GB
0,5 vCPU	1 GB
1 vCPU	2 GB
1 vCPU	3 GB
1 vCPU	4 GB
2 vCPU	4 GB
2 vCPU	6 GB
4 vCPU	8 GB
4 vCPU	10 GB
4 vCPU	12 GB

Sumber daya App Runner

Saat Anda menggunakan App Runner, Anda membuat dan mengelola beberapa jenis sumber daya di situs Anda Akun AWS. Sumber daya ini digunakan untuk mengakses kode Anda dan mengelola layanan Anda.

Tabel berikut memberikan ikhtisar sumber daya ini:

Nama sumber daya	Deskripsi
Service	Merupakan versi aplikasi yang sedang berjalan. Sebagian besar panduan ini menjelaskan jenis layanan, manajemen, konfigurasi, dan pemantauan. ARN: `arn:aws:apprunner:` *`region`*`:`*`account-id`*` :service/`*`service-name`*` [/`*`service-id`*` ]`

Nama sumber daya	Deskripsi
Connection	Menyediakan layanan App Runner Anda akses ke repositori pribadi yang disimpan dengan penyedia pihak ketiga. Ada sebagai sumber daya terpisah untuk berbagi di beberapa layanan. Untuk informasi selengkapnya tentang koneksi, lihatthe section called “Koneksi”. ARN: `arn:aws:apprunner: region:account-id :connection/ connection-name [/connection-id ]`
AutoScalingConfiguration	Menyediakan layanan App Runner Anda dengan pengaturan yang mengontrol penskalaan otomatis aplikasi Anda. Ada sebagai sumber daya terpisah untuk berbagi di beberapa layanan. Untuk informasi selengkapnya tentang penskalaan otomatis, lihat the section called “Penskalaan otomatis”. ARN: `arn:aws:apprunner: region:account-id :autoscalingconfiguration/ config-name [/config-revision [/config-id ]]`
ObservabilityConfiguration	Mengonfigurasi fitur observabilitas aplikasi tambahan untuk layanan App Runner Anda. Ada sebagai sumber daya terpisah untuk berbagi di beberapa layanan. Untuk informasi selengkapnya tentang konfigurasi observabilitas, lihatthe section called “Konfigurasi observabilitas”. ARN: `arn:aws:apprunner: region:account-id :observabilityconfiguration/ config-name [/config-revision [/config-id ]]`
VpcConnector	Mengonfigurasi setelan VPC untuk layanan App Runner Anda. Ada sebagai sumber daya terpisah untuk berbagi di beberapa layanan. Untuk informasi selengkapnya tentang fungsionalitas VPC, lihat. the section called “Lalu lintas keluar” ARN: `arn:aws:apprunner: region:account-id :vpcconnector/ connector-name [/connector-revision [/connector-id ]]`

Nama sumber daya	Deskripsi
VpcIngressConnection	Ini adalah AWS App Runner sumber daya yang digunakan untuk mengkonfigurasi lalu lintas masuk. Ini membuat koneksi antara titik akhir antarmuka VPC dan layanan App Runner, untuk membuat layanan App Runner Anda dapat diakses hanya dari dalam VPC Amazon. Untuk informasi selengkapnya tentang fungsionalitas VPCIngress Koneksi, lihatthe section called “Aktifkan titik akhir Pribadi”. ARN: `arn:aws:apprunner: region:account-id :vpcingressconnection/ vpc-ingress-connection-name [/connector-id ]]`

Kuota sumber daya App Runner

AWS memberlakukan beberapa kuota (juga dikenal sebagai batas) pada akun Anda untuk penggunaan AWS sumber daya di masing-masing. Wilayah AWS Tabel berikut mencantumkan kuota yang terkait dengan resource App Runner. Kuota juga tercantum dalam AWS App Runner titik akhir dan kuota di. Referensi Umum AWS

Kuota sumber daya		Deskripsi	Nilai default	Dapat disesuaikan?
Services		Jumlah maksimum layanan yang dapat Anda buat di akun Anda untuk masing-masing Wilayah AWS.	30	✓ Ya
Connections		Jumlah maksimum koneksi yang dapat Anda buat di akun Anda untuk masing-masing Wilayah AWS. Anda dapat menggunakan satu koneksi di beberapa layanan.	10	✓ Ya
Auto scaling configurations	nama	Jumlah maksimum nama unik yang dapat Anda miliki dalam konfigurasi penskalaan otomatis yang Anda buat di akun Anda untuk masing-masing. Wilayah AWS Anda dapat menggunak	10	✓ Ya

Kuota sumber daya		Deskripsi	Nilai default	Dapat disesuaikan?
		an konfigurasi penskalaan otomatis tunggal di beberapa layanan.		
	revisi per nama	Jumlah maksimum revisi konfigurasi penskalaan otomatis yang dapat Anda buat di akun Anda Wilayah AWS untuk masing-masing nama unik. Anda dapat menggunakan revisi konfigurasi penskalaan otomatis tunggal di beberapa layanan.	5	✕ Tidak
Observability configurations	nama	Jumlah maksimum nama unik yang dapat Anda miliki dalam konfigurasi observabilitas yang Anda buat di akun Anda untuk masing-masing. Wilayah AWS Anda dapat menggunakan konfigurasi observabilitas tunggal di beberapa layanan.	10	✓ Ya
	revisi per nama	Jumlah maksimum revisi konfigurasi observabilitas yang dapat Anda buat di akun Anda Wilayah AWS untuk masing-masing nama unik. Anda dapat menggunakan revisi konfigurasi observabilitas tunggal di beberapa layanan.	10	✕ Tidak
VPC connectors		Jumlah maksimum konektor VPC yang dapat Anda buat di akun Anda untuk masing-masing. Wilayah AWS Anda dapat menggunakan konektor VPC tunggal di beberapa layanan.	10	✓ Ya
VPC Ingress Connection		Jumlah maksimum Koneksi Ingress VPC yang dapat Anda buat di akun Anda untuk masing-masing. Wilayah AWS Anda dapat menggunakan satu Koneksi Ingress VPC untuk mengakses beberapa layanan App Runner.	1	✕ Tidak

Sebagian besar kuota dapat disesuaikan, dan Anda dapat meminta peningkatan kuota untuk mereka. Untuk informasi selengkapnya, lihat Permintaan peningkatan kuota dalam Panduan Pengguna Service Quotas.

Layanan App Runner berdasarkan gambar sumber

Anda dapat menggunakan AWS App Runner untuk membuat dan mengelola layanan berdasarkan dua jenis sumber layanan yang berbeda secara fundamental: kode sumber dan gambar sumber. Terlepas dari jenis sumbernya, App Runner menangani memulai, menjalankan, menskalakan, dan menyeimbangkan beban layanan Anda. Anda dapat menggunakan kemampuan CI/CD App Runner untuk melacak perubahan pada gambar sumber atau kode Anda. Ketika App Runner menemukan perubahan, itu secara otomatis membangun (untuk kode sumber) dan menerapkan versi baru ke layanan App Runner Anda.

Bab ini membahas layanan berdasarkan gambar sumber. Untuk informasi tentang layanan berdasarkan kode sumber, lihatLayanan berbasis kode.

Gambar sumber adalah gambar kontainer publik atau pribadi yang disimpan dalam repositori gambar. Anda mengarahkan App Runner ke gambar, dan itu memulai layanan yang menjalankan wadah berdasarkan gambar ini. Tidak diperlukan tahap pembangunan. Sebaliknya, Anda memberikan ready-to-deploy gambar.

> **Note**
>
> Saat memberikan gambar kontainer, Anda bertanggung jawab untuk memperbarui dan menambal gambar-gambar ini secara teratur. Sementara App Runner mengelola infrastruktur, Anda harus memastikan keamanan dan up-to-date status gambar kontainer yang disediakan. Untuk informasi selengkapnya, lihat AWS App Runner Documentation

Penyedia repositori gambar

App Runner mendukung penyedia repositori gambar berikut:

- Amazon Elastic Container Registry (Amazon ECR) - Menyimpan gambar yang bersifat pribadi ke file. Akun AWS
- Amazon Elastic Container Registry Public (Amazon ECR Public) - Menyimpan gambar yang dapat dibaca publik.

Kasus penggunaan penyedia

- Menggunakan gambar yang disimpan di Amazon ECR di akun Anda AWS
- Menggunakan gambar yang disimpan di Amazon ECR di akun yang berbeda AWS
- Menggunakan gambar yang disimpan di Amazon ECR Public

Menggunakan gambar yang disimpan di Amazon ECR di akun Anda AWS

Amazon ECR menyimpan gambar di repositori. Ada repositori pribadi dan publik. Untuk menerapkan gambar Anda ke layanan App Runner dari repositori pribadi, App Runner memerlukan izin untuk membaca gambar Anda dari Amazon ECR. Untuk memberikan izin tersebut ke App Runner, Anda harus memberikan peran akses kepada App Runner. Ini adalah peran AWS Identity and Access Management (IAM) yang memiliki izin tindakan Amazon ECR yang diperlukan. Saat menggunakan konsol App Runner untuk membuat layanan, Anda dapat memilih peran yang ada di akun Anda. Atau, Anda dapat menggunakan konsol IAM untuk membuat peran kustom baru. Atau, Anda dapat memilih konsol App Runner untuk membuat peran berdasarkan kebijakan terkelola.

Saat Anda menggunakan App Runner API atau API AWS CLI, Anda menyelesaikan proses dua langkah. Pertama, Anda menggunakan konsol IAM untuk membuat peran akses. Anda dapat menggunakan kebijakan terkelola yang disediakan oleh App Runner atau memasukkan izin khusus Anda sendiri. Kemudian, Anda memberikan peran akses selama pembuatan layanan menggunakan tindakan CreateServiceAPI.

Untuk informasi tentang pembuatan layanan App Runner, lihatthe section called “Pembuatan”.

Menggunakan gambar yang disimpan di Amazon ECR di akun yang berbeda AWS

Saat membuat layanan App Runner, Anda dapat menggunakan gambar yang disimpan di repositori Amazon ECR milik akun selain AWS akun tempat layanan Anda berada. Ada beberapa pertimbangan tambahan yang perlu diingat saat menggunakan gambar lintas akun, selain yang tercantum di bagian sebelumnya tentang gambar akun yang sama.

- Repositori lintas akun harus memiliki kebijakan yang melekat padanya. Kebijakan repositori menyediakan peran akses Anda dengan izin untuk membaca gambar di repositori. Gunakan kebijakan berikut untuk tujuan ini. Ganti `access-role-arn` dengan Nama Sumber Daya Amazon (ARN) peran akses Anda.

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "access-role-arn"
      },
      "Action": [
        "ecr:BatchGetImage",
        "ecr:DescribeImages",
        "ecr:GetDownloadUrlForLayer"
      ]
    }
  ]
}
```

Untuk informasi tentang melampirkan kebijakan repositori ke repositori Amazon ECR, lihat Menyetel pernyataan kebijakan repositori di Panduan Pengguna Registri Amazon Elastic Container.

- App Runner tidak mendukung penerapan otomatis untuk gambar Amazon ECR di akun yang berbeda dari akun yang digunakan layanan Anda.

Menggunakan gambar yang disimpan di Amazon ECR Public

Amazon ECR Public menyimpan gambar yang dapat dibaca publik. Ini adalah perbedaan utama antara Amazon ECR dan Amazon ECR Public yang harus Anda ketahui dalam konteks layanan App Runner:

- Amazon ECR Gambar publik dapat dibaca publik. Anda tidak perlu memberikan peran akses saat membuat layanan berdasarkan image Amazon ECR Public. Repositori tidak memerlukan kebijakan apa pun yang dilampirkan padanya.
- App Runner tidak mendukung penerapan otomatis (berkelanjutan) untuk gambar Amazon ECR Public.

Luncurkan layanan langsung dari Amazon ECR Public

Anda dapat langsung meluncurkan gambar kontainer dari aplikasi web yang kompatibel yang di-host di Galeri Publik Amazon ECR sebagai layanan web yang berjalan di App Runner. Saat menjelajahi galeri, cari Launch with App Runner di halaman galeri untuk mendapatkan gambar.

Gambar dengan opsi ini kompatibel dengan App Runner. Untuk informasi selengkapnya tentang galeri, lihat Menggunakan Galeri Publik Amazon ECR di panduan pengguna Amazon ECR Public.

Untuk meluncurkan gambar galeri sebagai layanan App Runner

1. Pada halaman galeri gambar, pilih Luncurkan dengan App Runner.

   Hasil: Konsol App Runner terbuka di tab browser baru. Konsol menampilkan wizard Buat layanan, dengan sebagian besar detail layanan baru yang diperlukan telah diisi sebelumnya.
2. Jika Anda ingin membuat layanan di AWS Wilayah selain yang ditampilkan konsol, pilih Wilayah yang ditampilkan di header konsol. Kemudian, pilih Wilayah lain.
3. Untuk Port, masukkan nomor port tempat aplikasi gambar mendengarkan. Anda biasanya dapat menemukannya di halaman galeri untuk gambar.
4. Secara opsional, ubah detail konfigurasi lainnya.
5. Pilih Berikutnya, tinjau pengaturan, lalu pilih Buat & terapkan.

Contoh gambar

Tim App Runner mempertahankan gambar hello-app-runnercontoh di Galeri Publik Amazon ECR. Anda dapat menggunakan contoh ini untuk memulai membuat layanan App Runner berbasis gambar. Untuk informasi selengkapnya, lihat hello-app-runner.

Layanan App Runner berdasarkan kode sumber

Anda dapat menggunakan AWS App Runner untuk membuat dan mengelola layanan berdasarkan dua jenis sumber layanan yang berbeda secara fundamental: kode sumber dan gambar sumber. Terlepas dari jenis sumbernya, App Runner menangani memulai, menjalankan, menskalakan, dan menyeimbangkan beban layanan Anda. Anda dapat menggunakan kemampuan CI/CD dari App Runner untuk melacak perubahan pada gambar sumber atau kode Anda. Ketika App Runner menemukan perubahan, itu secara otomatis membangun (untuk kode sumber) dan menerapkan versi baru ke layanan App Runner Anda.

Bab ini membahas layanan berdasarkan kode sumber. Untuk informasi tentang layanan berdasarkan gambar sumber, lihatLayanan berbasis gambar.

Kode sumber adalah kode aplikasi yang dibuat dan diterapkan oleh App Runner untuk Anda. Anda mengarahkan App Runner ke direktori sumber dalam repositori kode dan memilih runtime yang sesuai yang sesuai dengan versi platform pemrograman. App Runner membuat gambar yang didasarkan pada gambar dasar runtime dan kode aplikasi Anda. Kemudian memulai layanan yang menjalankan wadah berdasarkan gambar ini.

App Runner menyediakan runtime terkelola khusus platform yang nyaman. Masing-masing runtime ini membuat gambar kontainer dari kode sumber Anda, dan menambahkan dependensi runtime bahasa ke dalam gambar Anda. Anda tidak perlu menyediakan konfigurasi kontainer dan membangun instruksi seperti Dockerfile.

Subtopik dari bagian ini membahas berbagai platform yang didukung App Runner— platform terkelola yang menyediakan runtime terkelola untuk lingkungan dan versi pemrograman yang berbeda.

Topik

- Penyedia repositori kode sumber
- Direktori sumber
- Platform terkelola App Runner
- Versi runtime terkelola dan build App Runner
- Menggunakan platform Python
- Menggunakan platform Node.js
- Menggunakan platform Java

- Menggunakan platform.NET
- Menggunakan platform PHP
- Menggunakan platform Ruby
- Menggunakan platform Go

Penyedia repositori kode sumber

App Runner menyebarkan kode sumber Anda dengan membacanya dari repositori kode sumber. App Runner mendukung dua penyedia repositori kode sumber: GitHubdan Bitbucket.

Menerapkan dari penyedia repositori kode sumber Anda

Untuk menerapkan kode sumber Anda ke layanan App Runner dari repositori kode sumber, App Runner membuat koneksi ke sana. Saat Anda menggunakan konsol App Runner untuk membuat layanan, Anda memberikan detail koneksi dan direktori sumber untuk App Runner untuk menerapkan kode sumber Anda.

Koneksi

Anda memberikan detail koneksi sebagai bagian dari prosedur pembuatan layanan. Saat Anda menggunakan App Runner API atau API AWS CLI, koneksi adalah sumber daya terpisah. Pertama, Anda membuat koneksi menggunakan aksi CreateConnectionAPI. Kemudian, Anda menyediakan ARN koneksi selama pembuatan layanan menggunakan tindakan CreateServiceAPI.

Direktori sumber

Saat Anda membuat layanan, Anda juga menyediakan direktori sumber. Secara default, App Runner menggunakan direktori root repositori Anda sebagai direktori sumber. Direktori sumber adalah lokasi di repositori kode sumber Anda yang menyimpan kode sumber dan file konfigurasi aplikasi Anda. Perintah build dan start juga dijalankan dari direktori sumber. Bila Anda menggunakan App Runner API atau AWS CLI untuk membuat atau memperbarui layanan, Anda menyediakan direktori sumber dalam tindakan CreateServicedan UpdateServiceAPI. Untuk informasi lebih lanjut, lihat Direktori sumber bagian berikut.

Untuk informasi selengkapnya tentang pembuatan layanan App Runner, lihatthe section called “Pembuatan”. Untuk informasi selengkapnya tentang koneksi App Runner, lihatthe section called “Koneksi”.

Direktori sumber

Saat Anda membuat layanan App Runner, Anda dapat menyediakan direktori sumber, bersama dengan repositori dan cabang. Tetapkan nilai bidang direktori Sumber ke jalur direktori repositori yang menyimpan kode sumber dan file konfigurasi aplikasi. App Runner mengeksekusi perintah build dan start dari jalur direktori sumber yang Anda berikan.

Masukkan nilai untuk jalur direktori sumber sebagai absolut dari direktori repositori root. Jika Anda tidak menentukan nilai, itu default ke direktori tingkat atas repositori, juga dikenal sebagai direktori root repositori.

Anda juga memiliki opsi untuk menyediakan jalur direktori sumber yang berbeda selain direktori repositori tingkat atas. Ini mendukung arsitektur repositori monorepo, yang berarti kode sumber untuk beberapa aplikasi disimpan dalam satu repositori. Untuk membuat dan mendukung beberapa layanan App Runner dari satu monorepo, tentukan direktori sumber yang berbeda saat Anda membuat setiap layanan.

> **Note**
>
> Jika Anda menentukan direktori sumber yang sama untuk beberapa layanan App Runner, kedua layanan akan menerapkan dan beroperasi secara individual.

Jika Anda memilih untuk menggunakan file `apprunner.yaml` konfigurasi untuk menentukan parameter layanan Anda, letakkan di folder direktori sumber repositori.

Jika opsi pemicu Deployment disetel ke Otomatis, perubahan yang Anda lakukan di direktori sumber akan memicu penerapan otomatis. Hanya perubahan di jalur direktori sumber yang akan memicu penerapan otomatis. Penting untuk memahami bagaimana lokasi direktori sumber memengaruhi ruang lingkup penerapan otomatis. Untuk informasi selengkapnya, lihat penerapan otomatis di. Metode deployment

> **Note**
>
> Jika layanan App Runner Anda menggunakan runtime terkelola PHP, dan Anda ingin menetapkan direktori sumber selain repositori root default, penting untuk menggunakan versi runtime PHP yang benar. Untuk informasi selengkapnya, lihat Menggunakan platform PHP.

Platform terkelola App Runner

Platform terkelola App Runner menyediakan runtime terkelola untuk berbagai lingkungan pemrograman. Setiap runtime terkelola memudahkan untuk membangun dan menjalankan container berdasarkan versi bahasa pemrograman atau lingkungan runtime. Bila Anda menggunakan runtime terkelola, App Runner dimulai dengan image runtime terkelola. Gambar ini didasarkan pada image Amazon Linux Docker dan berisi paket runtime bahasa serta beberapa alat dan paket ketergantungan populer. App Runner menggunakan image runtime terkelola ini sebagai image dasar, dan menambahkan kode aplikasi Anda untuk membuat image Docker. Kemudian menyebarkan gambar ini untuk menjalankan layanan web Anda dalam sebuah wadah.

Anda menentukan runtime untuk layanan App Runner saat membuat layanan menggunakan konsol App Runner atau operasi API. CreateService Anda juga dapat menentukan runtime sebagai bagian dari kode sumber Anda. Gunakan `runtime` kata kunci dalam file konfigurasi App Runner yang Anda sertakan dalam repositori kode Anda. Konvensi penamaan dari runtime terkelola adalah*`<language-name><major-version>`*.

App Runner memperbarui runtime untuk layanan Anda ke versi terbaru pada setiap penerapan atau pembaruan layanan. Jika aplikasi Anda memerlukan versi tertentu dari runtime terkelola, Anda dapat menentukannya menggunakan `runtime-version` kata kunci dalam file konfigurasi App Runner. Anda dapat mengunci ke tingkat versi apa pun, termasuk versi mayor atau minor. App Runner hanya membuat pembaruan tingkat yang lebih rendah ke runtime layanan Anda.

Versi runtime terkelola dan build App Runner

App Runner menawarkan proses build yang diperbarui untuk aplikasi yang berjalan pada runtime versi utama yang lebih baru. Proses pembuatan yang direvisi ini lebih cepat dan lebih efisien. Ini juga membuat gambar akhir dengan footprint yang lebih kecil yang hanya berisi kode sumber Anda, membangun artefak, dan runtime yang diperlukan untuk menjalankan aplikasi Anda.

Kami menyebut proses build yang lebih baru sebagai build App Runner yang direvisi dan proses build asli sebagai build App Runner asli. Untuk menghindari perubahan pada platform runtime versi sebelumnya, App Runner hanya menerapkan build yang direvisi ke versi runtime tertentu, biasanya rilis utama yang baru dirilis.

Kami telah memperkenalkan komponen baru ke file `apprunner.yaml` konfigurasi untuk membuat build yang direvisi kompatibel ke belakang untuk kasus penggunaan yang sangat spesifik dan juga memberikan lebih banyak fleksibilitas untuk mengonfigurasi build aplikasi Anda. Ini adalah `pre-`

`run`parameter opsional. Kami menjelaskan kapan harus menggunakan parameter ini bersama dengan informasi berguna lainnya tentang build di bagian berikutnya.

Tabel berikut menunjukkan versi build App Runner mana yang berlaku untuk versi runtime terkelola tertentu. Kami akan terus memperbarui dokumen ini untuk memberi Anda informasi tentang runtime kami saat ini.

Platform	Build asli	Membangun yang direvisi
Python – Informasi rilis	• Python 3.8 • Python 3.7	• Python 3.11 (!)
Node.js – Informasi rilis	• Node.js 16 • Node.js 14 • Node.js 12	• Node.js 22 • Node.js 18
Corretto — Informasi rilis	• Corretto 11 • Corretto 8	
.NET – Informasi rilis	• .NET 6	
PHP – Informasi rilis	• PHP 8.1	
Ruby – Informasi rilis	• Ruby 3.1	
Go – Informasi rilis	• Pergi 1	

> **Important**
>
> Python 3.11 — Kami memiliki rekomendasi khusus untuk konfigurasi build layanan yang menggunakan runtime terkelola Python 3.11. Untuk informasi selengkapnya, lihat Callout untuk versi runtime tertentu di topik platform Python.

Selengkapnya tentang build dan migrasi App Runner

Saat memigrasikan aplikasi ke runtime yang lebih baru yang menggunakan build yang direvisi, Anda mungkin perlu sedikit mengubah konfigurasi build.

Untuk memberikan konteks pertimbangan migrasi, pertama-tama kami akan menjelaskan proses tingkat tinggi untuk build App Runner asli dan build yang direvisi. Kami akan mengikuti dengan bagian yang menjelaskan atribut spesifik tentang layanan Anda yang mungkin memerlukan beberapa pembaruan konfigurasi.

Versi App Runner asli

Proses pembuatan aplikasi App Runner asli memanfaatkan layanan. AWS CodeBuild Langkah awal didasarkan pada gambar yang dikuratori oleh CodeBuild layanan. Proses build Docker mengikuti yang menggunakan image runtime terkelola App Runner yang berlaku sebagai image dasar.

Langkah-langkah umum adalah sebagai berikut:

1. Jalankan `pre-build` perintah dalam gambar CodeBuild -curated.

   `pre-build`Perintahnya opsional. Mereka hanya dapat ditentukan dalam file `apprunner.yaml` konfigurasi.
2. Jalankan `build` perintah menggunakan CodeBuild pada gambar yang sama dari langkah sebelumnya.

   `build`Perintah diperlukan. Mereka dapat ditentukan di konsol App Runner, App Runner API, atau dalam file `apprunner.yaml` konfigurasi.
3. Jalankan build Docker untuk menghasilkan gambar berdasarkan image runtime terkelola App Runner untuk platform dan versi runtime spesifik Anda.
4. Salin `/app` direktori dari gambar yang kami hasilkan di Langkah 2. Tujuannya adalah gambar berdasarkan gambar runtime terkelola App Runner, yang kami buat di Langkah 3.
5. Jalankan `build` perintah lagi pada image runtime terkelola App Runner yang dihasilkan. Kami menjalankan perintah build lagi untuk menghasilkan artefak build dari kode sumber di `/app` direktori yang kami salin di Langkah 4. Gambar ini nantinya akan digunakan oleh App Runner untuk menjalankan layanan web Anda dalam wadah.

   `build`Perintah diperlukan. Mereka dapat ditentukan di konsol App Runner, App Runner API, atau dalam file `apprunner.yaml` konfigurasi.
6. Jalankan `post-build` perintah dalam CodeBuild gambar dari Langkah 2.

   `post-build`Perintahnya opsional. Mereka hanya dapat ditentukan dalam file `apprunner.yaml` konfigurasi.

Setelah build selesai, App Runner akan menerapkan image runtime terkelola App Runner yang dihasilkan dari Langkah 5 untuk menjalankan layanan web Anda dalam sebuah container.

Versi App Runner yang direvisi

Proses build yang direvisi lebih cepat dan lebih efisien daripada proses build asli yang dijelaskan di bagian sebelumnya. Ini menghilangkan duplikasi perintah build yang terjadi di build versi sebelumnya. Ini juga membuat gambar akhir dengan footprint yang lebih kecil yang hanya berisi kode sumber Anda, membangun artefak, dan runtime yang diperlukan untuk menjalankan aplikasi Anda.

Proses build ini menggunakan Docker multi-stage build. Langkah-langkah proses umum adalah sebagai berikut:

1. Build stage — Mulai proses docker build yang mengeksekusi `pre-build` dan `build` memerintahkan di atas image build App Runner.
   a. Salin kode sumber aplikasi ke `/app` direktori.

      > **Note**
      >
      > `/app`Direktori ini ditetapkan sebagai direktori kerja di setiap tahap build Docker.

   b. Jalankan perintah `pre-build`.

      `pre-build`Perintahnya opsional. Mereka hanya dapat ditentukan dalam file `apprunner.yaml` konfigurasi.

   c. Jalankan `build` perintah.

      `build`Perintah diperlukan. Mereka dapat ditentukan di konsol App Runner, App Runner API, atau dalam file `apprunner.yaml` konfigurasi.
2. Tahap pengemasan - Menghasilkan gambar kontainer pelanggan akhir, yang juga didasarkan pada image run App Runner.
   a. Salin `/app` direktori dari tahap Build sebelumnya ke image Run baru. Ini termasuk kode sumber aplikasi Anda dan artefak build dari tahap sebelumnya.
   b. Jalankan `pre-run` perintah. Jika Anda perlu memodifikasi gambar runtime di luar `/app` direktori dengan menggunakan `build` perintah, tambahkan perintah yang sama atau yang diperlukan ke segmen file `apprunner.yaml` konfigurasi ini.

Ini adalah parameter baru yang diperkenalkan untuk mendukung build App Runner yang direvisi.

`pre-run`Perintahnya opsional. Mereka hanya dapat ditentukan dalam file `apprunner.yaml` konfigurasi.

> **Catatan**
>
> - `pre-run`Perintah hanya didukung oleh build yang direvisi. Jangan menambahkannya ke file konfigurasi jika layanan Anda menggunakan versi runtime yang menggunakan build asli.
> - Jika Anda tidak perlu memodifikasi apa pun di luar `/app` direktori dengan `build` perintah, maka Anda tidak perlu menentukan `pre-run` perintah.

3. Tahap pasca-build - Tahap ini dilanjutkan dari tahap Build dan menjalankan perintah. `post-build`

   a. Jalankan `post-build` perintah di dalam `/app` direktori.

   `post-build`Perintahnya opsional. Mereka hanya dapat ditentukan dalam file `apprunner.yaml` konfigurasi.

Setelah build selesai, App Runner kemudian menerapkan image Run untuk menjalankan layanan web Anda dalam sebuah container.

> **Note**
>
> Jangan disesatkan ke `env` entri di bagian Jalankan `apprunner.yaml` saat mengonfigurasi proses pembuatan. Meskipun parameter `pre-run` perintah, yang direferensikan di Langkah 2 (b), berada di bagian Run, jangan gunakan `env` parameter di bagian Run untuk mengonfigurasi build Anda. `pre-run`Perintah hanya mereferensikan `env` variabel yang ditentukan di bagian Build dari file konfigurasi. Untuk informasi selengkapnya, lihat Jalankan bagian di bagian file konfigurasi App Runner.

Persyaratan layanan untuk pertimbangan migrasi

Jika lingkungan aplikasi Anda memiliki salah satu dari dua persyaratan ini, maka Anda harus merevisi konfigurasi build Anda, dengan menambahkan `pre-run` perintah.

- Jika Anda perlu memodifikasi apa pun di luar `/app` direktori dengan `build` perintah.
- Jika Anda perlu menjalankan `build` perintah dua kali untuk membuat lingkungan yang diperlukan. Ini adalah persyaratan yang sangat tidak biasa. Sebagian besar build tidak akan melakukan ini.

Modifikasi di luar **`/app`** direktori

- Build App Runner yang direvisi mengasumsikan bahwa aplikasi Anda tidak memiliki dependensi di luar direktori. `/app`
- Perintah yang Anda berikan dengan `apprunner.yaml` file, API App Runner, atau konsol App Runner harus menghasilkan artefak build di direktori. `/app`
- Anda dapat memodifikasi `post-build` perintah `pre-buildbuild`,, dan untuk memastikan semua artefak build ada di `/app` direktori.
- Jika aplikasi Anda memerlukan build untuk memodifikasi lebih lanjut gambar yang dihasilkan untuk layanan Anda, di luar `/app` direktori, Anda dapat menggunakan `pre-run` perintah baru di file`apprunner.yaml`. Untuk informasi selengkapnya, lihat Menyetel opsi layanan App Runner menggunakan file konfigurasi.

Menjalankan **`build`** perintah dua kali

- App Runner build asli menjalankan `build` perintah dua kali, pertama di Langkah 2, lalu lagi di Langkah 5. App Runner build yang direvisi memperbaiki redundansi ini dan hanya menjalankan perintah satu kali. `build` Jika aplikasi Anda harus memiliki persyaratan yang tidak biasa agar `build` perintah dijalankan dua kali, build App Runner yang direvisi menyediakan opsi untuk menentukan dan menjalankan perintah yang sama lagi menggunakan parameter. `pre-run` Melakukannya mempertahankan perilaku build ganda yang sama.

Menggunakan platform Python

Platform AWS App Runner Python menyediakan runtime terkelola. Setiap runtime memudahkan untuk membangun dan menjalankan container dengan aplikasi web berdasarkan versi Python. Saat Anda menggunakan runtime Python, App Runner dimulai dengan image runtime Python yang

dikelola. Gambar ini didasarkan pada image Amazon Linux Docker dan berisi paket runtime untuk versi Python dan beberapa alat dan paket ketergantungan populer. App Runner menggunakan image runtime terkelola ini sebagai image dasar, dan menambahkan kode aplikasi Anda untuk membuat image Docker. Kemudian menyebarkan gambar ini untuk menjalankan layanan web Anda dalam sebuah wadah.

Anda menentukan runtime untuk layanan App Runner saat membuat layanan menggunakan konsol App Runner atau operasi API. CreateService Anda juga dapat menentukan runtime sebagai bagian dari kode sumber Anda. Gunakan `runtime` kata kunci dalam file konfigurasi App Runner yang Anda sertakan dalam repositori kode Anda. Konvensi penamaan dari runtime terkelola adalah*<language-name><major-version>*.

Untuk nama dan versi runtime Python yang valid, lihat. the section called “Informasi rilis”

App Runner memperbarui runtime untuk layanan Anda ke versi terbaru pada setiap penerapan atau pembaruan layanan. Jika aplikasi Anda memerlukan versi tertentu dari runtime terkelola, Anda dapat menentukannya menggunakan `runtime-version` kata kunci dalam file konfigurasi App Runner. Anda dapat mengunci ke tingkat versi apa pun, termasuk versi mayor atau minor. App Runner hanya membuat pembaruan tingkat yang lebih rendah ke runtime layanan Anda.

Sintaks versi untuk runtime Python: *major*`[.`*minor*`[.`*patch*`]]`

Misalnya: `3.8.5`

Contoh berikut menunjukkan penguncian versi:

- `3.8`— Kunci versi mayor dan minor. App Runner hanya memperbarui versi tambalan.
- `3.8.5`— Kunci ke versi patch tertentu. App Runner tidak memperbarui versi runtime Anda.

Topik

- Konfigurasi runtime Python
- Callout untuk versi runtime tertentu
- Contoh runtime Python
- Informasi rilis runtime Python

Konfigurasi runtime Python

Saat memilih runtime terkelola, Anda juga harus mengonfigurasi, seminimal mungkin, membangun dan menjalankan perintah. Anda mengonfigurasinya saat membuat atau memperbarui layanan App Runner. Anda dapat melakukan ini menggunakan salah satu metode berikut:

- Menggunakan konsol App Runner — Tentukan perintah di bagian Configure build pada tab proses pembuatan atau konfigurasi.
- Menggunakan App Runner API — Panggil operasi CreateServiceatau UpdateServiceAPI. Tentukan perintah menggunakan `BuildCommand` dan `StartCommand` anggota tipe CodeConfigurationValuesdata.
- Menggunakan file konfigurasi — Tentukan satu atau beberapa perintah build hingga tiga fase build, dan satu perintah run yang berfungsi untuk memulai aplikasi Anda. Ada pengaturan konfigurasi opsional tambahan.

Menyediakan file konfigurasi adalah opsional. Saat membuat layanan App Runner menggunakan konsol atau API, Anda menentukan apakah App Runner mendapatkan setelan konfigurasi secara langsung saat dibuat atau dari file konfigurasi.

Callout untuk versi runtime tertentu

> **Note**
>
> App Runner sekarang menjalankan proses build yang diperbarui untuk aplikasi berdasarkan versi runtime berikut: Python 3.11, Node.js 22, dan Node.js 18. Jika aplikasi Anda berjalan pada salah satu versi runtime ini, lihat Versi runtime terkelola dan build App Runner untuk informasi selengkapnya tentang proses build yang direvisi. Aplikasi yang menggunakan semua versi runtime lainnya tidak terpengaruh, dan mereka terus menggunakan proses build asli.

Python 3.11 (build App Runner yang direvisi)

Gunakan pengaturan berikut di apprunner.yaml untuk runtime Python 3.11 yang dikelola.

- Atur `runtime` kunci di bagian Atas ke `python311`

Example

```
runtime: python311
```

- Gunakan `pip3` alih-alih `pip` untuk menginstal dependensi.
- Gunakan `python3` interpreter sebagai gantinya. `python`
- Jalankan `pip3` installer sebagai `pre-run` perintah. Python menginstal dependensi di luar direktori. `/app` Karena App Runner menjalankan build App Runner yang direvisi untuk Python 3.11, apa pun yang diinstal di luar `/app` direktori melalui perintah di bagian Build file akan hilang. `apprunner.yaml` Untuk informasi selengkapnya, lihat Versi App Runner yang direvisi.

  Example

```
run:
  runtime-version: 3.11
  pre-run:
    - pip3 install pipenv
    - pipenv install
    - python3 copy-global-files.py
  command: pipenv run gunicorn django_apprunner.wsgi --log-file -
```

Untuk informasi selengkapnya, lihat juga contoh file konfigurasi yang diperluas untuk Python 3.11 nanti dalam topik ini.

Contoh runtime Python

Contoh berikut menunjukkan file konfigurasi App Runner untuk membangun dan menjalankan layanan Python. Contoh terakhir adalah kode sumber untuk aplikasi Python lengkap yang dapat Anda gunakan ke layanan runtime Python.

> **Note**
>
> Versi runtime yang digunakan dalam contoh ini adalah *`3.7.7`* dan*`3.11`*. Anda dapat menggantinya dengan versi yang ingin Anda gunakan. Untuk versi runtime Python terbaru yang didukung, lihat. the section called “Informasi rilis”

File konfigurasi Python minimal

Contoh ini menunjukkan file konfigurasi minimal yang dapat Anda gunakan dengan runtime terkelola Python. Untuk asumsi yang dibuat oleh App Runner dengan file konfigurasi minimal, lihat. the section called “Contoh file konfigurasi”

Python 3.11 menggunakan perintah dan`pip3`. `python3` Untuk informasi selengkapnya, lihat contoh file konfigurasi yang diperluas untuk Python 3.11 nanti dalam topik ini.

Example apprunner.yaml

```
version: 1.0
runtime: python3
build:
  commands:
    build:
      - pip install pipenv
      - pipenv install
run:
  command: python app.py
```

File konfigurasi Python yang diperluas

Contoh ini menunjukkan penggunaan semua kunci konfigurasi dengan runtime terkelola Python.

> **Note**
>
> Versi runtime yang digunakan dalam contoh ini adalah`3.7.7`. Anda dapat menggantinya dengan versi yang ingin Anda gunakan. Untuk versi runtime Python terbaru yang didukung, lihat. the section called “Informasi rilis”
> Python 3.11 menggunakan perintah dan`pip3`. `python3` Untuk informasi selengkapnya, lihat contoh file konfigurasi yang diperluas untuk Python 3.11 nanti dalam topik ini.

Example apprunner.yaml

```
version: 1.0
runtime: python3
build:
  commands:
```

```
    pre-build:
      - wget -c https://s3.amazonaws.com/amzn-s3-demo-bucket/test-lib.tar.gz -O - | tar
 -xz
    build:
      - pip install pipenv
      - pipenv install
    post-build:
      - python manage.py test
  env:
    - name: DJANGO_SETTINGS_MODULE
      value: "django_apprunner.settings"
    - name: MY_VAR_EXAMPLE
      value: "example"
run:
  runtime-version: 3.7.7
  command: pipenv run gunicorn django_apprunner.wsgi --log-file -
  network:
    port: 8000
    env: MY_APP_PORT
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
  secrets:
    - name: my-secret
      value-from: "arn:aws:secretsmanager:us-
east-1:123456789012:secret:testingstackAppRunnerConstr-kJFXde2ULKbT-S7t8xR:username::"
    - name: my-parameter
      value-from: "arn:aws:ssm:us-east-1:123456789012:parameter/parameter-name"
    - name: my-parameter-only-name
      value-from: "parameter-name"
```

File konfigurasi Python yang diperluas - Python 3.11 (menggunakan build yang direvisi)

Contoh ini menunjukkan penggunaan semua kunci konfigurasi dengan runtime terkelola Python 3.11 di. `apprunner.yaml` Contoh ini menyertakan `pre-run` bagian, karena versi Python ini menggunakan build App Runner yang direvisi.

`pre-run`Parameter ini hanya didukung oleh build App Runner yang direvisi. Jangan masukkan parameter ini dalam file konfigurasi jika aplikasi Anda menggunakan versi runtime yang didukung oleh build App Runner asli. Untuk informasi selengkapnya, lihat Versi runtime terkelola dan build App Runner.

> **Note**
>
> Versi runtime yang digunakan dalam contoh ini adalah*3.11*. Anda dapat menggantinya dengan versi yang ingin Anda gunakan. Untuk versi runtime Python terbaru yang didukung, lihat. the section called “Informasi rilis”

Example apprunner.yaml

```
version: 1.0
runtime: python311
build:
  commands:
    pre-build:
      - wget -c https://s3.amazonaws.com/amzn-s3-demo-bucket/test-lib.tar.gz -O - | tar
 -xz
    build:
      - pip3 install pipenv
      - pipenv install
    post-build:
      - python3 manage.py test
  env:
    - name: DJANGO_SETTINGS_MODULE
      value: "django_apprunner.settings"
    - name: MY_VAR_EXAMPLE
      value: "example"
run:
  runtime-version: 3.11
  pre-run:
    - pip3 install pipenv
    - pipenv install
    - python3 copy-global-files.py
  command: pipenv run gunicorn django_apprunner.wsgi --log-file -
  network:
    port: 8000
    env: MY_APP_PORT
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
  secrets:
    - name: my-secret
```

```
      value-from: "arn:aws:secretsmanager:us-
east-1:123456789012:secret:testingstackAppRunnerConstr-kJFXde2ULKbT-S7t8xR:username::"
    - name: my-parameter
      value-from: "arn:aws:ssm:us-east-1:123456789012:parameter/parameter-name"
    - name: my-parameter-only-name
      value-from: "parameter-name"
```

Sumber aplikasi Python lengkap

Contoh ini menunjukkan kode sumber untuk aplikasi Python lengkap yang dapat Anda gunakan ke layanan runtime Python.

Example requirements.txt

```
pyramid==2.0
```

Example server.py

```
from wsgiref.simple_server import make_server
from pyramid.config import Configurator
from pyramid.response import Response
import os

def hello_world(request):
    name = os.environ.get('NAME')
    if name == None or len(name) == 0:
        name = "world"
    message = "Hello, " + name + "!\n"
    return Response(message)

if __name__ == '__main__':
    port = int(os.environ.get("PORT"))
    with Configurator() as config:
        config.add_route('hello', '/')
        config.add_view(hello_world, route_name='hello')
        app = config.make_wsgi_app()
    server = make_server('0.0.0.0', port, app)
    server.serve_forever()
```

Example apprunner.yaml

```
version: 1.0
```

```
runtime: python3
build:
  commands:
    build:
      - pip install -r requirements.txt
run:
  command: python server.py
```

Informasi rilis runtime Python

Topik ini mencantumkan detail lengkap untuk versi runtime Python yang didukung App Runner.

Versi runtime yang didukung — build App Runner yang direvisi

Nama runtime	Versi minor	Paket termasuk
Python 3.11 (python311)	3.11.12	SQLite 3.50.0
	3.11.11	SQLite 3.49.1
	3.11.10	SQLite 3.46.1
	3.11.9	SQLite 3.46.1
	3.11.8	SQLite 3.45.2
	3.11.7	SQLite 3.44.2

Catatan

- Python 3.11 — Kami memiliki rekomendasi khusus untuk konfigurasi build layanan yang menggunakan runtime terkelola Python 3.11. Untuk informasi selengkapnya, lihat Callout untuk versi runtime tertentu di topik platform Python.
- App Runner menyediakan proses build yang direvisi untuk runtime utama tertentu yang telah dirilis baru-baru ini. Karena itu, Anda akan melihat referensi ke build App Runner yang direvisi dan build App Runner asli di bagian tertentu dari dokumen ini. Untuk informasi selengkapnya, lihat Versi runtime terkelola dan build App Runner.

Versi runtime yang didukung — build App Runner asli

Nama runtime	Versi minor	Paket termasuk
Python 3 (python3)	3.8.20	SQLite 3.50.0
	3.8.16	SQLite 3.46.1
	3.8.15	SQLite 3.40.0
	3.7.16	SQLite 3.50.0
	3.7.15	SQLite 3.40.0
	3.7.10	SQLite 3.40.0

> **Note**
>
> App Runner menyediakan proses build yang direvisi untuk runtime utama tertentu yang telah dirilis baru-baru ini. Karena itu, Anda akan melihat referensi ke build App Runner yang direvisi dan build App Runner asli di bagian tertentu dari dokumen ini. Untuk informasi selengkapnya, lihat Versi runtime terkelola dan build App Runner.

Menggunakan platform Node.js

Platform AWS App Runner Node.js menyediakan runtime terkelola. Setiap runtime memudahkan untuk membangun dan menjalankan container dengan aplikasi web berdasarkan versi Node.js. Saat Anda menggunakan runtime Node.js, App Runner dimulai dengan image runtime Node.js yang dikelola. Gambar ini didasarkan pada image Amazon Linux Docker dan berisi paket runtime untuk versi Node.js dan beberapa alat. App Runner menggunakan image runtime terkelola ini sebagai image dasar, dan menambahkan kode aplikasi Anda untuk membuat image Docker. Kemudian menyebarkan gambar ini untuk menjalankan layanan web Anda dalam sebuah wadah.

Anda menentukan runtime untuk layanan App Runner saat membuat layanan menggunakan konsol App Runner atau operasi API. CreateService Anda juga dapat menentukan runtime sebagai bagian dari kode sumber Anda. Gunakan `runtime` kata kunci dalam file konfigurasi App Runner yang Anda sertakan dalam repositori kode Anda. Konvensi penamaan dari runtime terkelola adalah*<language-name><major-version>*.

Untuk nama dan versi runtime Node.js yang valid, lihatthe section called “Informasi rilis”.

App Runner memperbarui runtime untuk layanan Anda ke versi terbaru pada setiap penerapan atau pembaruan layanan. Jika aplikasi Anda memerlukan versi tertentu dari runtime terkelola, Anda dapat menentukannya menggunakan `runtime-version` kata kunci dalam file konfigurasi App Runner. Anda dapat mengunci ke tingkat versi apa pun, termasuk versi mayor atau minor. App Runner hanya membuat pembaruan tingkat yang lebih rendah ke runtime layanan Anda.

Sintaks versi untuk runtime Node.js: `major[.minor[.patch]]`

Misalnya: `22.14.0`

Contoh berikut menunjukkan penguncian versi:

- `22.14`— Kunci versi mayor dan minor. App Runner hanya memperbarui versi tambalan.
- `22.14.0`— Kunci ke versi patch tertentu. App Runner tidak memperbarui versi runtime Anda.

Topik

- Konfigurasi runtime Node.js
- Callout untuk versi runtime tertentu
- Contoh runtime Node.js
- Informasi rilis runtime Node.js

Konfigurasi runtime Node.js

Saat memilih runtime terkelola, Anda juga harus mengonfigurasi, seminimal mungkin, membangun dan menjalankan perintah. Anda mengonfigurasinya saat membuat atau memperbarui layanan App Runner. Anda dapat melakukan ini menggunakan salah satu metode berikut:

- Menggunakan konsol App Runner — Tentukan perintah di bagian Configure build pada tab proses pembuatan atau konfigurasi.
- Menggunakan App Runner API — Panggil operasi CreateServiceatau UpdateServiceAPI. Tentukan perintah menggunakan `BuildCommand` dan `StartCommand` anggota tipe CodeConfigurationValuesdata.
- Menggunakan file konfigurasi — Tentukan satu atau beberapa perintah build hingga tiga fase build, dan satu perintah run yang berfungsi untuk memulai aplikasi Anda. Ada pengaturan konfigurasi opsional tambahan.

Menyediakan file konfigurasi adalah opsional. Saat membuat layanan App Runner menggunakan konsol atau API, Anda menentukan apakah App Runner mendapatkan setelan konfigurasi secara langsung saat dibuat atau dari file konfigurasi.

Dengan runtime Node.js secara khusus, Anda juga dapat mengonfigurasi build dan runtime menggunakan file JSON yang diberi nama `package.json` di root repositori sumber Anda. Dengan menggunakan file ini, Anda dapat mengonfigurasi versi mesin Node.js, paket ketergantungan, dan berbagai perintah (aplikasi baris perintah). Package manager seperti npm atau yarn menafsirkan file ini sebagai masukan untuk perintah mereka.

Misalnya:

- npm installmenginstal paket yang ditentukan oleh `dependencies` dan `devDependencies` node di`package.json`.
- npm startatau npm run start menjalankan perintah yang ditentukan oleh `scripts/start` node di`package.json`.

Berikut ini adalah contoh `package.json` file.

package.json

```
{
  "name": "node-js-getting-started",
  "version": "0.3.0",
  "description": "A sample Node.js app using Express 4",
  "engines": {
    "node": "22.14.0"
  },
  "scripts": {
    "start": "node index.js",
    "test": "node test.js"
  },
  "dependencies": {
    "cool-ascii-faces": "^1.3.4",
    "ejs": "^2.5.6",
    "express": "^4.15.2"
  },
  "devDependencies": {
    "got": "^11.3.0",
    "tape": "^4.7.0"
  }
```

```
}
```

Untuk informasi selengkapnya*`package.json`*, lihat [Membuat file package.json](#) di situs web npm Docs.

Kiat

- Jika `package.json` file Anda mendefinisikan start perintah, Anda dapat menggunakannya sebagai run perintah dalam file konfigurasi App Runner, seperti yang ditunjukkan contoh berikut.

  Example

  package.json

  ```
  {
    "scripts": {
      "start": "node index.js"
    }
  }
  ```

  apprunner.yaml

  ```
  run:
    command: npm start
  ```

- Ketika Anda menjalankan npm install di lingkungan pengembangan Anda, npm membuat file`package-lock.json`. File ini berisi snapshot dari versi paket npm yang baru saja diinstal. Setelah itu, ketika npm menginstal dependensi, ia menggunakan versi yang tepat ini. Jika Anda menginstal benang, itu akan membuat `yarn.lock` file. Komit file-file ini ke repositori kode sumber Anda untuk memastikan bahwa aplikasi Anda diinstal dengan versi dependensi yang Anda kembangkan dan uji.
- Anda juga dapat menggunakan file konfigurasi App Runner untuk mengonfigurasi versi Node.js dan memulai perintah. Ketika Anda melakukan ini, definisi ini mengesampingkan yang masuk`package.json`. Konflik antara `node` versi masuk `package.json` dan `runtime-version` nilai dalam file konfigurasi App Runner menyebabkan fase build App Runner gagal.

Callout untuk versi runtime tertentu

Node.js 22 dan Node.js 18 (build App Runner yang direvisi)

App Runner sekarang menjalankan proses build yang diperbarui untuk aplikasi berdasarkan versi runtime berikut: Python 3.11, Node.js 22, dan Node.js 18. Jika aplikasi Anda berjalan pada salah satu versi runtime ini, lihat Versi runtime terkelola dan build App Runner untuk informasi selengkapnya tentang proses build yang direvisi. Aplikasi yang menggunakan semua versi runtime lainnya tidak terpengaruh, dan mereka terus menggunakan proses build asli.

Contoh runtime Node.js

Contoh berikut menunjukkan file konfigurasi App Runner untuk membangun dan menjalankan layanan Node.js.

> **Note**
>
> Versi runtime yang digunakan dalam contoh ini adalah`22.14.0`. Anda dapat menggantinya dengan versi yang ingin Anda gunakan. Untuk versi runtime Node.js terbaru yang didukung, lihatthe section called “Informasi rilis”.

File konfigurasi Node.js minimal

Contoh ini menunjukkan file konfigurasi minimal yang dapat Anda gunakan dengan runtime terkelola Node.js. Untuk asumsi yang dibuat oleh App Runner dengan file konfigurasi minimal, lihat. the section called “Contoh file konfigurasi”

Example apprunner.yaml

```
version: 1.0
runtime: nodejs22
build:
  commands:
    build:
      - npm install --production
run:
  command: node app.js
```

File konfigurasi Node.js yang diperluas

Contoh ini menunjukkan penggunaan semua kunci konfigurasi dengan runtime terkelola Node.js.

> **Note**
>
> Versi runtime yang digunakan dalam contoh ini adalah*22.14.0*. Anda dapat menggantinya dengan versi yang ingin Anda gunakan. Untuk versi runtime Node.js terbaru yang didukung, lihatthe section called “Informasi rilis”.

Example apprunner.yaml

```
version: 1.0
runtime: nodejs22
build:
  commands:
    pre-build:
      - npm install --only=dev
      - node test.js
    build:
      - npm install --production
    post-build:
      - node node_modules/ejs/postinstall.js
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
run:
  runtime-version: 22.14.0
  command: node app.js
  network:
    port: 8000
    env: APP_PORT
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
```

File konfigurasi Node.js yang diperluas - Node.js 22 (menggunakan build yang direvisi)

Contoh ini menunjukkan penggunaan semua kunci konfigurasi dengan runtime terkelola Node.js di. `apprunner.yaml` Contoh ini menyertakan `pre-run` bagian, karena versi Node.js ini menggunakan build App Runner yang direvisi.

`pre-run`Parameter ini hanya didukung oleh build App Runner yang direvisi. Jangan masukkan parameter ini dalam file konfigurasi jika aplikasi Anda menggunakan versi runtime yang didukung oleh build App Runner asli. Untuk informasi selengkapnya, lihat Versi runtime terkelola dan build App Runner.

> **Note**
>
> Versi runtime yang digunakan dalam contoh ini adalah*22.14.0*. Anda dapat menggantinya dengan versi yang ingin Anda gunakan. Untuk versi runtime Node.js terbaru yang didukung, lihatthe section called “Informasi rilis”.

Example apprunner.yaml

```
version: 1.0
runtime: nodejs22
build:
  commands:
    pre-build:
      - npm install --only=dev
      - node test.js
    build:
      - npm install --production
    post-build:
      - node node_modules/ejs/postinstall.js
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
run:
  runtime-version: 22.14.0
  pre-run:
    - node copy-global-files.js
  command: node app.js
  network:
    port: 8000
    env: APP_PORT
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
```

Aplikasi Node.js dengan Grunt

Contoh ini menunjukkan cara mengkonfigurasi aplikasi Node.js yang dikembangkan dengan Grunt. Grunt adalah runner JavaScript tugas baris perintah. Ini menjalankan tugas berulang dan mengelola otomatisasi proses untuk mengurangi kesalahan manusia. Plugin Grunt dan Grunt diinstal dan dikelola menggunakan npm. Anda mengkonfigurasi Grunt dengan memasukkan `Gruntfile.js` file di root repositori sumber Anda.

Example package.json

```
{
  "scripts": {
    "build": "grunt uglify",
    "start": "node app.js"
  },
  "devDependencies": {
    "grunt": "~0.4.5",
    "grunt-contrib-jshint": "~0.10.0",
    "grunt-contrib-nodeunit": "~0.4.1",
    "grunt-contrib-uglify": "~0.5.0"
  },
  "dependencies": {
    "express": "^4.15.2"
  },
}
```

Example Gruntfile.js

```
module.exports = function(grunt) {

  // Project configuration.
  grunt.initConfig({
    pkg: grunt.file.readJSON('package.json'),
    uglify: {
      options: {
        banner: '/*! <%= pkg.name %> <%= grunt.template.today("yyyy-mm-dd") %> */\n'
      },
      build: {
        src: 'src/<%= pkg.name %>.js',
        dest: 'build/<%= pkg.name %>.min.js'
      }
    }
  });
```

```
  // Load the plugin that provides the "uglify" task.
  grunt.loadNpmTasks('grunt-contrib-uglify');

  // Default task(s).
  grunt.registerTask('default', ['uglify']);

};
```

Example apprunner.yaml

> **Note**
>
> Versi runtime yang digunakan dalam contoh ini adalah*22.14.0*. Anda dapat menggantinya dengan versi yang ingin Anda gunakan. Untuk versi runtime Node.js terbaru yang didukung, lihatthe section called “Informasi rilis”.

```
version: 1.0
runtime: nodejs22
build:
  commands:
    pre-build:
      - npm install grunt grunt-cli
      - npm install --only=dev
      - npm run build
    build:
      - npm install --production
run:
  runtime-version: 22.14.0
  command: node app.js
  network:
    port: 8000
    env: APP_PORT
```

Informasi rilis runtime Node.js

> **Note**
>
> Kebijakan penghentian standar App Runner adalah menghentikan runtime ketika komponen utama runtime mencapai akhir dukungan jangka panjang komunitas (LTS) dan pembaruan

keamanan tidak lagi tersedia. Dalam beberapa kasus, App Runner dapat menunda penghentian runtime untuk periode terbatas, di luar end-of-support tanggal versi bahasa yang didukung oleh runtime. Contoh kasus seperti itu adalah memperluas dukungan untuk runtime guna memungkinkan waktu pelanggan untuk migrasi.

Topik ini mencantumkan detail lengkap untuk versi runtime Node.js yang didukung App Runner.

Versi runtime yang didukung — build App Runner yang direvisi

Nama runtime	Versi minor	Paket termasuk
Node.js 22 (nodejs22)	22.16.0	npm 10.9.2, benang 1.22.22
Node.js 22 (nodejs22)	22.14.0	npm 10.9.2, benang 1.22.22

> **Note**
>
> App Runner menyediakan proses build yang direvisi untuk runtime utama tertentu yang telah dirilis baru-baru ini. Karena itu, Anda akan melihat referensi ke build App Runner yang direvisi dan build App Runner asli di bagian tertentu dari dokumen ini. Untuk informasi selengkapnya, lihat Versi runtime terkelola dan build App Runner.

Versi runtime yang didukung — build App Runner yang direvisi

Nama runtime	Versi minor	Paket termasuk
Node.js 18 (nodejs18)	18.20.8	npm 10.8.2, benang 1.22.22
	18.20.7	npm 10.8.2, benang 1.22.22
	18.20.6	npm 10.8.2, benang 1.22.22
	18.20.5	npm 10.8.2, benang 1.22.22
	18.20.4	npm 10.7.0, benang 1.22.22
	18.20.3	npm 10.7.0, benang 1.22.22

Nama runtime	Versi minor	Paket termasuk
	18.20.2	npm 10, benang *
	18.19.1	npm 10, benang *
	18.19.0	npm 10, benang *

Versi runtime yang didukung — build App Runner asli

Nama runtime	Versi minor	Paket termasuk
Node.js 16 (nodejs16)	16.20.2	npm 8.19.4, benang 1.22.22
	16.20.1	npm 8.19.4, benang *
	16.20.0	npm 8.19.4, benang *
	16.19.1	npm 8.19.4, benang *
	16.19.0	npm 8.19.4, benang *
	16.18.1	npm 8.19.4, benang *
	16.17.1	npm 8.19.4, benang *
	16.17.0	npm 8.19.4, benang *
Node.js 14 (nodejs14)	14.21.3	npm 6.14.18, benang 1.22.22
	14.21.2	npm 6.14.18, benang *
	14.21.1	npm 6.14.18, benang *
	14.20.1	npm 6.14.18, benang *
	14.19.0	npm 6.14.18, benang *
Node.js 12 (nodejs12)	12.22.12	npm 6.14.16, benang 1.22.22

Nama runtime	Versi minor	Paket termasuk
	12.21.0	npm 6.14.16, benang *

Menggunakan platform Java

Platform AWS App Runner Java menyediakan runtime terkelola. Setiap runtime memudahkan untuk membangun dan menjalankan kontainer dengan aplikasi web berdasarkan versi Java. Saat Anda menggunakan runtime Java, App Runner dimulai dengan image runtime Java yang dikelola. Gambar ini didasarkan pada image Amazon Linux Docker dan berisi paket runtime untuk versi Java dan beberapa alat. App Runner menggunakan image runtime terkelola ini sebagai image dasar, dan menambahkan kode aplikasi Anda untuk membuat image Docker. Kemudian menyebarkan gambar ini untuk menjalankan layanan web Anda dalam sebuah wadah.

Anda menentukan runtime untuk layanan App Runner saat membuat layanan menggunakan konsol App Runner atau operasi API. CreateService Anda juga dapat menentukan runtime sebagai bagian dari kode sumber Anda. Gunakan `runtime` kata kunci dalam file konfigurasi App Runner yang Anda sertakan dalam repositori kode Anda. Konvensi penamaan dari runtime terkelola adalah*<language-name><major-version>*.

Saat ini, semua runtime Java yang didukung didasarkan pada Amazon Corretto. Untuk nama dan versi runtime Java yang valid, lihatthe section called “Informasi rilis”.

App Runner memperbarui runtime untuk layanan Anda ke versi terbaru pada setiap penerapan atau pembaruan layanan. Jika aplikasi Anda memerlukan versi tertentu dari runtime terkelola, Anda dapat menentukannya menggunakan `runtime-version` kata kunci dalam file konfigurasi App Runner. Anda dapat mengunci ke tingkat versi apa pun, termasuk versi mayor atau minor. App Runner hanya membuat pembaruan tingkat yang lebih rendah ke runtime layanan Anda.

Sintaks versi untuk runtime Amazon Corretto:

Runtime	Sintaksis	Contoh
corretto11	`11.0[.`*`openjdk-update`*` [.`*`openjdk-build`*` [.`*`corretto-specific-revision`*` ]]]`	`11.0.13.08.1`

Runtime	Sintaksis	Contoh
corretto8	`8[.openjdk-update [.openjdk-build [.corretto-specific-revision ]]]`	`8.312.07.1`

Contoh berikut menunjukkan penguncian versi:

- `11.0.13`— Kunci versi pembaruan JDK Terbuka. App Runner hanya memperbarui Open JDK dan Amazon Corretto build tingkat rendah.
- `11.0.13.08.1`— Kunci ke versi tertentu. App Runner tidak memperbarui versi runtime Anda.

Topik

- Konfigurasi runtime Java
- Contoh runtime Java
- Informasi rilis runtime Java

Konfigurasi runtime Java

Saat memilih runtime terkelola, Anda juga harus mengonfigurasi, seminimal mungkin, membangun dan menjalankan perintah. Anda mengonfigurasinya saat membuat atau memperbarui layanan App Runner. Anda dapat melakukan ini menggunakan salah satu metode berikut:

- Menggunakan konsol App Runner — Tentukan perintah di bagian Configure build pada tab proses pembuatan atau konfigurasi.
- Menggunakan App Runner API — Panggil operasi CreateServiceatau UpdateServiceAPI. Tentukan perintah menggunakan `BuildCommand` dan `StartCommand` anggota tipe CodeConfigurationValuesdata.
- Menggunakan file konfigurasi — Tentukan satu atau beberapa perintah build hingga tiga fase build, dan satu perintah run yang berfungsi untuk memulai aplikasi Anda. Ada pengaturan konfigurasi opsional tambahan.

Menyediakan file konfigurasi adalah opsional. Saat membuat layanan App Runner menggunakan konsol atau API, Anda menentukan apakah App Runner mendapatkan setelan konfigurasi secara langsung saat dibuat atau dari file konfigurasi.

Contoh runtime Java

Contoh berikut menunjukkan file konfigurasi App Runner untuk membangun dan menjalankan layanan Java. Contoh terakhir adalah kode sumber untuk aplikasi Java lengkap yang dapat Anda gunakan ke layanan runtime Corretto 11.

> **Note**
>
> Versi runtime yang digunakan dalam contoh ini adalah`11.0.13.08.1`. Anda dapat menggantinya dengan versi yang ingin Anda gunakan. Untuk versi runtime Java terbaru yang didukung, lihatthe section called “Informasi rilis”.

File konfigurasi Corretto 11 minimal

Contoh ini menunjukkan file konfigurasi minimal yang dapat Anda gunakan dengan runtime terkelola Corretto 11. Untuk asumsi yang dibuat oleh App Runner dengan file konfigurasi minimal, lihat.

Example apprunner.yaml

```
version: 1.0
runtime: corretto11
build:
  commands:
    build:
      - mvn clean package
run:
  command: java -Xms256m -jar target/MyApp-1.0-SNAPSHOT.jar .
```

File konfigurasi Corretto 11 yang diperluas

Contoh ini menunjukkan bagaimana Anda dapat menggunakan semua kunci konfigurasi dengan runtime terkelola Corretto 11.

> **Note**
>
> Versi runtime yang digunakan dalam contoh ini adalah*11.0.13.08.1*. Anda dapat menggantinya dengan versi yang ingin Anda gunakan. Untuk versi runtime Java terbaru yang didukung, lihatthe section called “Informasi rilis”.

Example apprunner.yaml

```
version: 1.0
runtime: corretto11
build:
  commands:
    pre-build:
      - yum install some-package
      - scripts/prebuild.sh
    build:
      - mvn clean package
    post-build:
      - mvn clean test
  env:
    - name: M2
      value: "/usr/local/apache-maven/bin"
    - name: M2_HOME
      value: "/usr/local/apache-maven/bin"
run:
  runtime-version: 11.0.13.08.1
  command: java -Xms256m -jar target/MyApp-1.0-SNAPSHOT.jar .
  network:
    port: 8000
    env: APP_PORT
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
```

Sumber aplikasi Corretto 11 lengkap

Contoh ini menunjukkan kode sumber untuk aplikasi Java lengkap yang dapat Anda gunakan ke layanan runtime Corretto 11.

Example src/main/java/com/HelloWorld/HelloWorld.jawa

```
package com.HelloWorld;

import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HelloWorld {

    @RequestMapping("/")
    public String index(){
        String s = "Hello World";
        return s;
    }
}
```

Example src/main/java/com/HelloWorld/Main.jawa

```
package com.HelloWorld;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Main {

    public static void main(String[] args) {

        SpringApplication.run(Main.class, args);
    }
}
```

Example apprunner.yaml

```
version: 1.0
runtime: corretto11
build:
  commands:
    build:
      - mvn clean package
run:
```

```
  command: java -Xms256m -jar target/HelloWorldJavaApp-1.0-SNAPSHOT.jar .
  network:
    port: 8080
```

Example pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/
maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.3.1.RELEASE</version>
    <relativePath/>
  </parent>
  <groupId>com.HelloWorld</groupId>
  <artifactId>HelloWorldJavaApp</artifactId>
  <version>1.0-SNAPSHOT</version>

  <properties>
    <java.version>11</java.version>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-rest</artifactId>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
      <exclusions>
        <exclusion>
          <groupId>org.junit.vintage</groupId>
          <artifactId>junit-vintage-engine</artifactId>
        </exclusion>
      </exclusions>
    </dependency>
```

```
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
      </plugin>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.0</version>
        <configuration>
          <release>11</release>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

Informasi rilis runtime Java

Topik ini mencantumkan detail lengkap untuk versi runtime Java yang didukung App Runner.

Versi runtime yang didukung — build App Runner asli

Nama runtime	Versi minor	Paket termasuk
Corretto 11 (Corretto11)	11.0.27.6.1	Maven 3.9.9, Gradle 6.9.4
	11.0.26.4.1	Maven 3.9.9, Gradle 6.9.4
	11.0.25.9.1	Maven 3.9.9, Gradle 6.9.4
	11.0.24.8.1	Maven 3.9.9, Gradle 6.9.4
	11.0.23.9.1	Maven 3.9.8, Gradle 6.9.4
	11.0.22.7.1	Maven 3.9.6, Gradle 6.9.4
	11.0.21.9.1	Maven 3.9.6, Gradle 6.9.4
	11.0.21.9.1	Maven 3.9.5, Gradle 6.9.4

Nama runtime	Versi minor	Paket termasuk
	11.0.20.8.1	Maven 3.9.3, Gradle 6.9.4
	11.0.19.7.1	Maven 3.9.3, Gradle 6.9.4
	11.0.18.10.1	Maven 3.9.1, Gradle 6.9.4
	11.0.17.8.1	Maven 3.8.6, Gradle 6.9.3
	11.0.16.9.1	Maven 3.8.6, Gradle 6.9.2
	11.0.13.08.1	Maven 3.6.3, Gradle 6.5
Corretto 8 (corretto8)	8.452.09.2	Maven 3.9.9, Gradle 6.9.4
	8.452.09.1	Maven 3.9.9, Gradle 6.9.4
	8.442.06.1	Maven 3.9.9, Gradle 6.9.4
	8.432.06.1	Maven 3.9.9, Gradle 6.9.4
	8.422.05.1	Maven 3.9.9, Gradle 6.9.4
	8.412.08.1	Maven 3.9.8, Gradle 6.9.4
	8.402.08.1	Maven 3.9.6, Gradle 6.9.4
	8.392.08.1	Maven 3.9.6, Gradle 6.9.4
	8.382.05.1	Maven 3.9.4, Gradle 6.9.4
	8.372.07.1	Maven 3.9.3, Gradle 6.9.4
	8.362.08.1	Maven 3.9.1, Gradle 6.9.4
	8.352.08.1	Maven 3.8.6, Gradle 6.9.3
	8.342.07.4	Maven 3.8.6, Gradle 6.9.2
	8.312.07.1	Maven 3.6.3, Gradle 6.5

> **Note**
>
> App Runner menyediakan proses build yang direvisi untuk runtime utama tertentu yang telah dirilis baru-baru ini. Karena itu, Anda akan melihat referensi ke build App Runner yang direvisi dan build App Runner asli di bagian tertentu dari dokumen ini. Untuk informasi selengkapnya, lihat Versi runtime terkelola dan build App Runner.

Menggunakan platform.NET

AWS App Runner Platform.NET menyediakan runtime terkelola. Setiap runtime memudahkan untuk membangun dan menjalankan container dengan aplikasi web berdasarkan versi.NET. Saat Anda menggunakan runtime.NET, App Runner dimulai dengan image runtime.NET yang dikelola. Gambar ini didasarkan pada image Amazon Linux Docker dan berisi paket runtime untuk versi.NET dan beberapa alat dan paket ketergantungan populer. App Runner menggunakan image runtime terkelola ini sebagai image dasar, dan menambahkan kode aplikasi Anda untuk membuat image Docker. Kemudian menyebarkan gambar ini untuk menjalankan layanan web Anda dalam sebuah wadah.

Anda menentukan runtime untuk layanan App Runner saat membuat layanan menggunakan konsol App Runner atau operasi API. CreateService Anda juga dapat menentukan runtime sebagai bagian dari kode sumber Anda. Gunakan `runtime` kata kunci dalam file konfigurasi App Runner yang Anda sertakan dalam repositori kode Anda. Konvensi penamaan dari runtime terkelola adalah*<language-name><major-version>*.

Untuk nama dan versi runtime .NET yang valid, lihatthe section called “Informasi rilis”.

App Runner memperbarui runtime untuk layanan Anda ke versi terbaru pada setiap penerapan atau pembaruan layanan. Jika aplikasi Anda memerlukan versi tertentu dari runtime terkelola, Anda dapat menentukannya menggunakan `runtime-version` kata kunci dalam file konfigurasi App Runner. Anda dapat mengunci ke tingkat versi apa pun, termasuk versi mayor atau minor. App Runner hanya membuat pembaruan tingkat yang lebih rendah ke runtime layanan Anda.

Sintaks versi untuk runtime.NET: *major*`[.`*minor*`[.`*patch*`]]`

Misalnya: `6.0.9`

Contoh berikut menunjukkan penguncian versi:

- `6.0`— Kunci versi mayor dan minor. App Runner hanya memperbarui versi tambalan.

- `6.0.9`— Kunci ke versi patch tertentu. App Runner tidak memperbarui versi runtime Anda.

Topik

- Konfigurasi runtime .NET
- Contoh runtime .NET
- .NET informasi rilis runtime

Konfigurasi runtime .NET

Saat memilih runtime terkelola, Anda juga harus mengonfigurasi, seminimal mungkin, membangun dan menjalankan perintah. Anda mengonfigurasinya saat membuat atau memperbarui layanan App Runner. Anda dapat melakukan ini menggunakan salah satu metode berikut:

- Menggunakan konsol App Runner — Tentukan perintah di bagian Configure build pada tab proses pembuatan atau konfigurasi.
- Menggunakan App Runner API — Panggil operasi CreateServiceatau UpdateServiceAPI. Tentukan perintah menggunakan `BuildCommand` dan `StartCommand` anggota tipe CodeConfigurationValuesdata.
- Menggunakan file konfigurasi — Tentukan satu atau beberapa perintah build hingga tiga fase build, dan satu perintah run yang berfungsi untuk memulai aplikasi Anda. Ada pengaturan konfigurasi opsional tambahan.

Menyediakan file konfigurasi adalah opsional. Saat membuat layanan App Runner menggunakan konsol atau API, Anda menentukan apakah App Runner mendapatkan setelan konfigurasi secara langsung saat dibuat atau dari file konfigurasi.

Contoh runtime .NET

Contoh berikut menunjukkan file konfigurasi App Runner untuk membangun dan menjalankan layanan.NET. Contoh terakhir adalah kode sumber untuk aplikasi.NET lengkap yang dapat Anda gunakan ke layanan runtime .NET.

> **Note**
>
> Versi runtime yang digunakan dalam contoh ini adalah`6.0.9`. Anda dapat menggantinya dengan versi yang ingin Anda gunakan. Untuk versi runtime.NET terbaru yang didukung, lihatthe section called “Informasi rilis”.

File konfigurasi.NET minimal

Contoh ini menunjukkan file konfigurasi minimal yang dapat Anda gunakan dengan runtime terkelola .NET. Untuk asumsi yang dibuat oleh App Runner dengan file konfigurasi minimal, lihat. the section called “Contoh file konfigurasi”

Example apprunner.yaml

```
version: 1.0
runtime: dotnet6
build:
  commands:
    build:
      - dotnet publish -c Release -o out
run:
  command: dotnet out/HelloWorldDotNetApp.dll
```

File konfigurasi.NET yang diperluas

Contoh ini menunjukkan penggunaan semua kunci konfigurasi dengan runtime terkelola .NET.

> **Note**
>
> Versi runtime yang digunakan dalam contoh ini adalah`6.0.9`. Anda dapat menggantinya dengan versi yang ingin Anda gunakan. Untuk versi runtime.NET terbaru yang didukung, lihatthe section called “Informasi rilis”.

Example apprunner.yaml

```
version: 1.0
runtime: dotnet6
build:
```

```
  commands:
    pre-build:
      - scripts/prebuild.sh
    build:
      - dotnet publish -c Release -o out
    post-build:
      - scripts/postbuild.sh
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
run:
  runtime-version: 6.0.9
  command: dotnet out/HelloWorldDotNetApp.dll
  network:
    port: 5000
    env: APP_PORT
  env:
    - name: ASPNETCORE_URLS
      value: "http://*:5000"
```

Sumber aplikasi.NET lengkap

Contoh ini menunjukkan kode sumber untuk aplikasi.NET lengkap yang dapat Anda gunakan ke layanan runtime .NET.

> **Note**
>
> - Jalankan perintah berikut untuk membuat aplikasi web.NET 6 sederhana: `dotnet new web --name HelloWorldDotNetApp -f net6.0`
> - Tambahkan `apprunner.yaml` ke aplikasi web.NET 6 yang dibuat.

Example HelloWorldDotNetApp

```
version: 1.0
runtime: dotnet6
build:
  commands:
    build:
      - dotnet publish -c Release -o out
run:
```

```
  command: dotnet out/HelloWorldDotNetApp.dll
  network:
    port: 5000
    env: APP_PORT
  env:
    - name: ASPNETCORE_URLS
      value: "http://*:5000"
```

.NET informasi rilis runtime

Topik ini mencantumkan detail lengkap untuk versi runtime.NET yang didukung App Runner.

Versi runtime yang didukung — build App Runner asli

Nama runtime	Versi minor	Paket termasuk
.NET 6 (dotnet6)	6.0.36	.NET SDK 6.0.428
	6.0.33	.NET SDK 6.0.425
	6.0.32	.NET SDK 6.0.424
	6.0.31	.NET SDK 6.0.423
	6.0.30	.NET SDK 6.0.422
	6.0.29	.NET SDK 6.0.421
	6.0.28	.NET SDK 6.0.420
	6.0.26	.NET SDK 6.0.418
	6.0.25	.NET SDK 6.0.417
	6.0.24	.NET SDK 6.0.416
	6.0.22	.NET SDK 6.0.414
	6.0.21	.NET SDK 6.0.413
	6.0.20	.NET SDK 6.0.412
	6.0.19	.NET SDK 6.0.411

Nama runtime	Versi minor	Paket termasuk
	6.0.16	.NET SDK 6.0.408
	6.0.15	.NET SDK 6.0.407
	6.0.14	.NET SDK 6.0.406
	6.0.13	.NET SDK 6.0.405
	6.0.12	.NET SDK 6.0.404
	6.0.11	.NET SDK 6.0.403
	6.0.10	.NET SDK 6.0.402
	6.0.9	.NET SDK 6.0.401

> **Note**
>
> App Runner menyediakan proses build yang direvisi untuk runtime utama tertentu yang telah dirilis baru-baru ini. Karena itu, Anda akan melihat referensi ke build App Runner yang direvisi dan build App Runner asli di bagian tertentu dari dokumen ini. Untuk informasi selengkapnya, lihat Versi runtime terkelola dan build App Runner.

Menggunakan platform PHP

Platform AWS App Runner PHP menyediakan runtime terkelola. Anda dapat menggunakan setiap runtime untuk membangun dan menjalankan container dengan aplikasi web berdasarkan versi PHP. Saat Anda menggunakan runtime PHP, App Runner dimulai dengan image runtime PHP yang dikelola. Gambar ini didasarkan pada image Amazon Linux Docker dan berisi paket runtime untuk versi PHP dan beberapa alat. App Runner menggunakan image runtime terkelola ini sebagai image dasar, dan menambahkan kode aplikasi Anda untuk membuat image Docker. Kemudian menyebarkan gambar ini untuk menjalankan layanan web Anda dalam sebuah wadah.

Anda menentukan runtime untuk layanan App Runner saat membuat layanan menggunakan konsol App Runner atau operasi API. CreateService Anda juga dapat menentukan runtime sebagai bagian

dari kode sumber Anda. Gunakan `runtime` kata kunci dalam file konfigurasi App Runner yang Anda sertakan dalam repositori kode Anda. Konvensi penamaan dari runtime terkelola adalah*<language-name><major-version>*.

Untuk nama dan versi runtime PHP yang valid, lihatthe section called “Informasi rilis”.

App Runner memperbarui runtime untuk layanan Anda ke versi terbaru pada setiap penerapan atau pembaruan layanan. Jika aplikasi Anda memerlukan versi tertentu dari runtime terkelola, Anda dapat menentukannya menggunakan `runtime-version` kata kunci dalam file konfigurasi App Runner. Anda dapat mengunci ke tingkat versi apa pun, termasuk versi mayor atau minor. App Runner hanya membuat pembaruan tingkat yang lebih rendah ke runtime layanan Anda.

Sintaks versi untuk runtime PHP: *major*`[.`*minor*`[.`*patch*`]]`

Misalnya: `8.1.10`

Berikut ini adalah contoh penguncian versi:

- `8.1`— Kunci versi mayor dan minor. App Runner hanya memperbarui versi tambalan.
- `8.1.10`— Kunci ke versi patch tertentu. App Runner tidak memperbarui versi runtime Anda.

> **Important**
>
> Jika Anda ingin menentukan direktori sumber repositori kode untuk layanan App Runner Anda di lokasi selain direktori root repositori default, versi runtime terkelola PHP Anda harus PHP atau yang lebih baru. `8.1.22` Versi runtime PHP sebelumnya hanya `8.1.22` dapat menggunakan direktori sumber root default.

Topik

- Konfigurasi runtime PHP
- Kompatibilitas
- Contoh runtime PHP
- Informasi rilis runtime PHP

Konfigurasi runtime PHP

Saat memilih runtime terkelola, Anda juga harus mengonfigurasi, seminimal mungkin, membangun dan menjalankan perintah. Anda mengonfigurasinya saat membuat atau memperbarui layanan App Runner. Anda dapat melakukan ini menggunakan salah satu metode berikut:

- Menggunakan konsol App Runner — Tentukan perintah di bagian Configure build pada tab proses pembuatan atau konfigurasi.
- Menggunakan App Runner API — Panggil operasi CreateServiceatau UpdateServiceAPI. Tentukan perintah menggunakan `BuildCommand` dan `StartCommand` anggota tipe CodeConfigurationValuesdata.
- Menggunakan file konfigurasi — Tentukan satu atau beberapa perintah build hingga tiga fase build, dan satu perintah run yang berfungsi untuk memulai aplikasi Anda. Ada pengaturan konfigurasi opsional tambahan.

Menyediakan file konfigurasi adalah opsional. Saat membuat layanan App Runner menggunakan konsol atau API, Anda menentukan apakah App Runner mendapatkan setelan konfigurasi secara langsung saat dibuat atau dari file konfigurasi.

Kompatibilitas

Anda dapat menjalankan layanan App Runner di platform PHP menggunakan salah satu server web berikut:

- Apache HTTP Server
- NGINX

Apache HTTP Server and NGINX kompatibel dengan PHP-FPM. Anda dapat memulai Apache HTTP Server dan NGINXdengan menggunakan salah satu dari berikut ini:

- Supervisord - Untuk informasi lebih lanjut tentang menjalankan supervisord, lihat Running supervisord.
- Skrip startup

Untuk contoh tentang cara mengonfigurasi layanan App Runner Anda dengan platform PHP menggunakan Apache HTTP Server atau NGINX, lihat. the section called “Sumber aplikasi PHP lengkap”

Struktur File

`index.php`Harus diinstal di `public` folder di bawah `root` direktori server web.

> **Note**
>
> Kami menyarankan agar `supervisord.conf` file `startup.sh` atau disimpan di direktori root server web. Pastikan bahwa `start` perintah menunjuk ke lokasi di mana `supervisord.conf` file `startup.sh` atau disimpan.

Berikut ini adalah contoh struktur file jika Anda menggunakan supervisord.

```
/
## public/
#  ## index.php
## apprunner.yaml
## supervisord.conf
```

Berikut ini adalah contoh struktur file jika Anda menggunakan skrip startup.

```
/
## public/
#  ## index.php
## apprunner.yaml
## startup.sh
```

Sebaiknya simpan struktur file ini di direktori sumber repositori kode yang ditujukan untuk layanan App Runner.

```
/<sourceDirectory>/
## public/
#  ## index.php
## apprunner.yaml
## startup.sh
```

> **Important**
>
> Jika Anda ingin menentukan direktori sumber repositori kode untuk layanan App Runner Anda di lokasi selain direktori root repositori default, versi runtime terkelola PHP Anda harus PHP atau yang lebih baru. `8.1.22` Versi runtime PHP sebelumnya hanya `8.1.22` dapat menggunakan direktori sumber root default.
> App Runner memperbarui runtime untuk layanan Anda ke versi terbaru pada setiap penerapan atau pembaruan layanan. Layanan Anda akan menggunakan runtime terbaru secara default, kecuali Anda menentukan penguncian versi menggunakan `runtime-version` kata kunci dalam file konfigurasi App Runner.

Contoh runtime PHP

Berikut ini adalah contoh file konfigurasi App Runner yang digunakan untuk membangun dan menjalankan layanan PHP.

File konfigurasi PHP minimal

Contoh berikut adalah file konfigurasi minimal yang dapat Anda gunakan dengan runtime terkelola PHP. Untuk informasi selengkapnya tentang file konfigurasi minimal, lihatthe section called “Contoh file konfigurasi”.

Example apprunner.yaml

```
version: 1.0
runtime: php81
build:
  commands:
    build:
      - echo example build command for PHP
run:
  command: ./startup.sh
```

File konfigurasi PHP yang diperluas

Contoh berikut menggunakan semua kunci konfigurasi dengan runtime terkelola PHP.

> **Note**
>
> Versi runtime yang digunakan dalam contoh ini adalah*8.1.10*. Anda dapat menggantinya dengan versi yang ingin Anda gunakan. Untuk versi runtime PHP terbaru yang didukung, lihatthe section called “Informasi rilis”.

Example apprunner.yaml

```
version: 1.0
runtime: php81
build:
  commands:
     pre-build:
      - scripts/prebuild.sh
    build:
      - echo example build command for PHP
    post-build:
      - scripts/postbuild.sh
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
run:
  runtime-version: 8.1.10
  command: ./startup.sh
  network:
    port: 5000
    env: APP_PORT
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
```

Sumber aplikasi PHP lengkap

Contoh berikut adalah kode sumber aplikasi PHP yang dapat Anda gunakan untuk menyebarkan ke layanan runtime PHP menggunakan Apache HTTP Server atau NGINX. Contoh-contoh ini mengasumsikan bahwa Anda menggunakan struktur file default.

Menjalankan platform PHP dengan Apache HTTP Server memakai supervisord

Example Struktur file

> **Note**
>
> - `supervisord.conf`File dapat disimpan di mana saja di repositori. Pastikan bahwa `start` perintah menunjuk ke tempat `supervisord.conf` file disimpan.
> - `index.php`Harus diinstal di `public` folder di bawah `root` direktori.

```
/
## public/
#  ## index.php
## apprunner.yaml
## supervisord.conf
```

Example supervisord.conf

```
[supervisord]
nodaemon=true

[program:httpd]
command=httpd -DFOREGROUND
autostart=true
autorestart=true
stdout_logfile=/dev/stdout
stdout_logfile_maxbytes=0
stderr_logfile=/dev/stderr
stderr_logfile_maxbytes=0

[program:php-fpm]
command=php-fpm -F
autostart=true
autorestart=true
stdout_logfile=/dev/stdout
stdout_logfile_maxbytes=0
stderr_logfile=/dev/stderr
stderr_logfile_maxbytes=0
```

Example apprunner.yaml

```
version: 1.0
runtime: php81
build:
  commands:
    build:
      - PYTHON=python2 amazon-linux-extras install epel
      - yum -y install supervisor
run:
  command: supervisord
  network:
    port: 8080
    env: APP_PORT
```

Example index.php

```
<html>
<head> <title>First PHP App</title> </head>
<body>
<?php
    print("Hello World!");
    print("<br>");
?>
</body>
</html>
```

Menjalankan platform PHP dengan Apache HTTP Server memakai startup script

Example Struktur file

> **Note**
>
> - `startup.sh`File dapat disimpan di mana saja di repositori. Pastikan bahwa `start` perintah menunjuk ke tempat `startup.sh` file disimpan.
> - `index.php`Harus diinstal di `public` folder di bawah `root` direktori.

```
/
## public/
```

```
#  ## index.php
## apprunner.yaml
## startup.sh
```

Example startup.sh

```
#!/bin/bash

set -o monitor

trap exit SIGCHLD

# Start apache
httpd -DFOREGROUND &

# Start php-fpm
php-fpm -F &

wait
```

> **Note**
>
> - Pastikan untuk menyimpan `startup.sh` file sebagai executable sebelum Anda mengkomitmennya ke repositori Git. Gunakan `chmod +x startup.sh` untuk mengatur izin eksekusi pada `startup.sh` file Anda.
> - Jika Anda tidak menyimpan `startup.sh` file sebagai executable, masukkan `chmod +x startup.sh` sebagai `build` perintah dalam file Anda`apprunner.yaml`.

Example apprunner.yaml

```
version: 1.0
runtime: php81
build:
  commands:
    build:
      - echo example build command for PHP
run:
  command: ./startup.sh
  network:
```

```
    port: 8080
    env: APP_PORT
```

Example index.php

```
<html>
<head> <title>First PHP App</title> </head>
<body>
<?php
    print("Hello World!");
    print("<br>");
?>
</body>
</html>
```

Menjalankan platform PHP dengan NGINX memakai supervisord

Example Struktur file

> **Note**
>
> - `supervisord.conf`File dapat disimpan di mana saja di repositori. Pastikan bahwa `start` perintah menunjuk ke tempat `supervisord.conf` file disimpan.
> - `index.php`Harus diinstal di `public` folder di bawah `root` direktori.

```
/
## public/
#  ## index.php
## apprunner.yaml
## supervisord.conf
```

Example supervisord.conf

```
[supervisord]
nodaemon=true

[program:nginx]
command=nginx -g "daemon off;"
```

```
autostart=true
autorestart=true
stdout_logfile=/dev/stdout
stdout_logfile_maxbytes=0
stderr_logfile=/dev/stderr
stderr_logfile_maxbytes=0

[program:php-fpm]
command=php-fpm -F
autostart=true
autorestart=true
stdout_logfile=/dev/stdout
stdout_logfile_maxbytes=0
stderr_logfile=/dev/stderr
stderr_logfile_maxbytes=0
```

Example apprunner.yaml

```
version: 1.0
runtime: php81
build:
  commands:
    build:
      - PYTHON=python2 amazon-linux-extras install epel
      - yum -y install supervisor
run:
  command: supervisord
  network:
    port: 8080
    env: APP_PORT
```

Example index.php

```
<html>
<head> <title>First PHP App</title> </head>
<body>
<?php
    print("Hello World!");
    print("<br>");
?>
</body>
</html>
```

Menjalankan platform PHP dengan NGINX memakai startup script

Example Struktur file

> **Note**
>
> - `startup.sh`File dapat disimpan di mana saja di repositori. Pastikan bahwa `start` perintah menunjuk ke tempat `startup.sh` file disimpan.
> - `index.php`Harus diinstal di `public` folder di bawah `root` direktori.

```
/
## public/
#  ## index.php
## apprunner.yaml
## startup.sh
```

Example startup.sh

```
#!/bin/bash

set -o monitor

trap exit SIGCHLD

# Start nginx
nginx -g 'daemon off;' &

# Start php-fpm
php-fpm -F &

wait
```

> **Note**
>
> - Pastikan untuk menyimpan `startup.sh` file sebagai executable sebelum Anda mengkomitmennya ke repositori Git. Gunakan `chmod +x startup.sh` untuk mengatur izin eksekusi pada `startup.sh` file Anda.

- Jika Anda tidak menyimpan `startup.sh` file sebagai executable, masukkan `chmod +x startup.sh` sebagai `build` perintah dalam file Anda`apprunner.yaml`.

Example apprunner.yaml

```
version: 1.0
runtime: php81
build:
  commands:
    build:
      - echo example build command for PHP
run:
  command: ./startup.sh
  network:
    port: 8080
    env: APP_PORT
```

Example index.php

```
<html>
<head> <title>First PHP App</title> </head>
<body>
<?php
    print("Hello World!");
    print("<br>");
?>
</body>
</html>
```

Informasi rilis runtime PHP

Topik ini mencantumkan detail lengkap untuk versi runtime PHP yang didukung App Runner.

Versi runtime yang didukung — build App Runner asli

Nama runtime	Versi minor	Paket termasuk
PHP 8.1 (php81)	8.1.32	
	8.1.31	

Nama runtime	Versi minor	Paket termasuk
	8.1.29	
	8.1.28	
	8.1.27	
	8.1.26	
	8.1.24	
	8.1.22	
	8.1.21	
	8.1.20	
	8.1.19	
	8.1.17	
	8.1.16	
	8.1.14	
	8.1.13	
	8.1.12	
	8.1.10	

> **Note**
>
> App Runner menyediakan proses build yang direvisi untuk runtime utama tertentu yang telah dirilis baru-baru ini. Karena itu, Anda akan melihat referensi ke build App Runner yang direvisi dan build App Runner asli di bagian tertentu dari dokumen ini. Untuk informasi selengkapnya, lihat Versi runtime terkelola dan build App Runner.

Menggunakan platform Ruby

Platform AWS App Runner Ruby menyediakan runtime terkelola. Setiap runtime memudahkan untuk membangun dan menjalankan container dengan aplikasi web berdasarkan versi Ruby. Saat Anda menggunakan runtime Ruby, App Runner dimulai dengan gambar runtime Ruby yang dikelola. Gambar ini didasarkan pada image Amazon Linux Docker dan berisi paket runtime untuk versi Ruby dan beberapa alat. App Runner menggunakan image runtime terkelola ini sebagai image dasar, dan menambahkan kode aplikasi Anda untuk membuat image Docker. Kemudian menyebarkan gambar ini untuk menjalankan layanan web Anda dalam sebuah wadah.

Anda menentukan runtime untuk layanan App Runner saat membuat layanan menggunakan konsol App Runner atau operasi API. CreateService Anda juga dapat menentukan runtime sebagai bagian dari kode sumber Anda. Gunakan `runtime` kata kunci dalam file konfigurasi App Runner yang Anda sertakan dalam repositori kode Anda. Konvensi penamaan runtime terkelola adalah*`<language-name><major-version>`*.

Untuk nama dan versi runtime Ruby yang valid, lihat. the section called “Informasi rilis”

App Runner memperbarui runtime untuk layanan Anda ke versi terbaru pada setiap penerapan atau pembaruan layanan. Jika aplikasi Anda memerlukan versi tertentu dari runtime terkelola, Anda dapat menentukannya menggunakan `runtime-version` kata kunci dalam file konfigurasi App Runner. Anda dapat mengunci ke tingkat versi apa pun, termasuk versi mayor atau minor. App Runner hanya membuat pembaruan tingkat yang lebih rendah ke runtime layanan Anda.

Sintaks versi untuk runtime Ruby: *`major`*`[.`*`minor`*`[.`*`patch`*`]]`

Misalnya: `3.1.2`

Contoh berikut menunjukkan penguncian versi:

- `3.1`— Kunci versi mayor dan minor. App Runner hanya memperbarui versi tambalan.
- `3.1.2`— Kunci ke versi patch tertentu. App Runner tidak memperbarui versi runtime Anda.

Topik

- Konfigurasi runtime Ruby
- Contoh runtime Ruby
- Informasi rilis runtime Ruby

Konfigurasi runtime Ruby

Saat memilih runtime terkelola, Anda juga harus mengonfigurasi, seminimal mungkin, membangun dan menjalankan perintah. Anda mengonfigurasinya saat membuat atau memperbarui layanan App Runner. Anda dapat melakukan ini menggunakan salah satu metode berikut:

- Menggunakan konsol App Runner — Tentukan perintah di bagian Configure build pada tab proses pembuatan atau konfigurasi.
- Menggunakan App Runner API — Panggil operasi CreateServiceatau UpdateServiceAPI. Tentukan perintah menggunakan `BuildCommand` dan `StartCommand` anggota tipe CodeConfigurationValuesdata.
- Menggunakan file konfigurasi — Tentukan satu atau beberapa perintah build hingga tiga fase build, dan satu perintah run yang berfungsi untuk memulai aplikasi Anda. Ada pengaturan konfigurasi opsional tambahan.

Menyediakan file konfigurasi adalah opsional. Saat membuat layanan App Runner menggunakan konsol atau API, Anda menentukan apakah App Runner mendapatkan setelan konfigurasi secara langsung saat dibuat atau dari file konfigurasi.

Contoh runtime Ruby

Contoh berikut menunjukkan file konfigurasi App Runner untuk membangun dan menjalankan layanan Ruby.

File konfigurasi Ruby minimal

Contoh ini menunjukkan file konfigurasi minimal yang dapat Anda gunakan dengan runtime terkelola Ruby. Untuk asumsi yang dibuat oleh App Runner dengan file konfigurasi minimal, lihat. the section called “Contoh file konfigurasi”

Example apprunner.yaml

```
version: 1.0
runtime: ruby31
build:
  commands:
    build:
      - bundle install
run:
```

```
  command: bundle exec rackup --host 0.0.0.0 -p 8080
```

File konfigurasi Ruby yang diperluas

Contoh ini menunjukkan penggunaan semua kunci konfigurasi dengan runtime terkelola Ruby.

> **Note**
>
> Versi runtime yang digunakan dalam contoh ini adalah*3.1.2*. Anda dapat menggantinya dengan versi yang ingin Anda gunakan. Untuk versi runtime Ruby terbaru yang didukung, lihat. the section called “Informasi rilis”

Example apprunner.yaml

```
version: 1.0
runtime: ruby31
build:
  commands:
     pre-build:
      - scripts/prebuild.sh
    build:
      - bundle install
    post-build:
      - scripts/postbuild.sh
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
run:
  runtime-version: 3.1.2
  command: bundle exec rackup --host 0.0.0.0 -p 4567
  network:
    port: 4567
    env: APP_PORT
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
```

Sumber aplikasi Ruby lengkap

Contoh-contoh ini menunjukkan kode sumber untuk aplikasi Ruby lengkap yang dapat Anda gunakan ke layanan runtime Ruby.

Example server.rb

```
# server.rb
require 'sinatra'

get '/' do
  'Hello World!'
end
```

Example config.ru

```
# config.ru

require './server'

run Sinatra::Application
```

Example Gemfile

```
# Gemfile
source 'https://rubygems.org (https://rubygems.org/)'

gem 'sinatra'
gem 'puma'
```

Example apprunner.yaml

```
version: 1.0
runtime: ruby31
build:
  commands:
    build:
      - bundle install
run:
  command: bundle exec rackup --host 0.0.0.0 -p 4567
  network:
    port: 4567
    env: APP_PORT
```

Informasi rilis runtime Ruby

Topik ini mencantumkan detail lengkap untuk versi runtime Ruby yang didukung App Runner.

Versi runtime yang didukung — build App Runner asli

Nama runtime	Versi minor	Paket termasuk
Ruby 3.1 (Ruby31)	3.1.7	SQLite 3.50.0
	3.1.6	SQLite 3.49.1
	3.1.4	SQLite 3.46.0
	3.1.3	SQLite 3.41.0
	3.1.2	SQLite 3.39.4

> **Note**
>
> App Runner menyediakan proses build yang direvisi untuk runtime utama tertentu yang telah dirilis baru-baru ini. Karena itu, Anda akan melihat referensi ke build App Runner yang direvisi dan build App Runner asli di bagian tertentu dari dokumen ini. Untuk informasi selengkapnya, lihat Versi runtime terkelola dan build App Runner.

Menggunakan platform Go

Platform AWS App Runner Go menyediakan runtime terkelola. Setiap runtime memudahkan untuk membangun dan menjalankan container dengan aplikasi web berdasarkan versi Go. Saat Anda menggunakan runtime Go, App Runner dimulai dengan image runtime Go yang dikelola. Gambar ini didasarkan pada image Amazon Linux Docker dan berisi paket runtime untuk versi Go dan beberapa alat. App Runner menggunakan image runtime terkelola ini sebagai image dasar, dan menambahkan kode aplikasi Anda untuk membuat image Docker. Kemudian menyebarkan gambar ini untuk menjalankan layanan web Anda dalam sebuah wadah.

Anda menentukan runtime untuk layanan App Runner saat membuat layanan menggunakan konsol App Runner atau operasi API. CreateService Anda juga dapat menentukan runtime sebagai bagian dari kode sumber Anda. Gunakan `runtime` kata kunci dalam file konfigurasi App Runner yang Anda sertakan dalam repositori kode Anda. Konvensi penamaan dari runtime terkelola adalah*<language-name><major-version>*.

Untuk nama dan versi runtime Go yang valid, lihatthe section called “Informasi rilis”.

App Runner memperbarui runtime untuk layanan Anda ke versi terbaru pada setiap penerapan atau pembaruan layanan. Jika aplikasi Anda memerlukan versi tertentu dari runtime terkelola, Anda dapat menentukannya menggunakan `runtime-version` kata kunci dalam file konfigurasi App Runner. Anda dapat mengunci ke tingkat versi apa pun, termasuk versi mayor atau minor. App Runner hanya membuat pembaruan tingkat yang lebih rendah ke runtime layanan Anda.

Sintaks versi untuk runtime Go: `major[.minor[.patch]]`

Misalnya: `1.18.7`

Contoh berikut menunjukkan penguncian versi:

- `1.18`— Kunci versi mayor dan minor. App Runner hanya memperbarui versi tambalan.
- `1.18.7`— Kunci ke versi patch tertentu. App Runner tidak memperbarui versi runtime Anda.

Topik

- Konfigurasi runtime Go
- Contoh runtime Go
- Informasi rilis Go runtime

Konfigurasi runtime Go

Saat memilih runtime terkelola, Anda juga harus mengonfigurasi, seminimal mungkin, membangun dan menjalankan perintah. Anda mengonfigurasinya saat membuat atau memperbarui layanan App Runner. Anda dapat melakukan ini menggunakan salah satu metode berikut:

- Menggunakan konsol App Runner — Tentukan perintah di bagian Configure build pada tab proses pembuatan atau konfigurasi.
- Menggunakan App Runner API — Panggil operasi CreateServiceatau UpdateServiceAPI. Tentukan perintah menggunakan `BuildCommand` dan `StartCommand` anggota tipe CodeConfigurationValuesdata.
- Menggunakan file konfigurasi — Tentukan satu atau beberapa perintah build hingga tiga fase build, dan satu perintah run yang berfungsi untuk memulai aplikasi Anda. Ada pengaturan konfigurasi opsional tambahan.

Menyediakan file konfigurasi adalah opsional. Saat membuat layanan App Runner menggunakan konsol atau API, Anda menentukan apakah App Runner mendapatkan setelan konfigurasi secara langsung saat dibuat atau dari file konfigurasi.

Contoh runtime Go

Contoh berikut menunjukkan file konfigurasi App Runner untuk membangun dan menjalankan layanan Go.

File konfigurasi Go Minimal

Contoh ini menunjukkan file konfigurasi minimal yang dapat Anda gunakan dengan runtime yang dikelola Go. Untuk asumsi yang dibuat oleh App Runner dengan file konfigurasi minimal, lihat. the section called “Contoh file konfigurasi”

Example apprunner.yaml

```
version: 1.0
runtime: go1
build:
  commands:
    build:
      - go build main.go
run:
  command: ./main
```

File konfigurasi Go yang diperluas

Contoh ini menunjukkan penggunaan semua kunci konfigurasi dengan runtime yang dikelola Go.

> **Note**
>
> Versi runtime yang digunakan dalam contoh ini adalah`1.18.7`. Anda dapat menggantinya dengan versi yang ingin Anda gunakan. Untuk versi runtime Go terbaru yang didukung, lihatthe section called “Informasi rilis”.

Example apprunner.yaml

```
version: 1.0
```

```
runtime: go1
build:
  commands:
     pre-build:
      - scripts/prebuild.sh
    build:
      - go build main.go
    post-build:
      - scripts/postbuild.sh
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
run:
  runtime-version: 1.18.7
  command: ./main
  network:
    port: 3000
    env: APP_PORT
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
```

Sumber aplikasi Go lengkap

Contoh-contoh ini menunjukkan kode sumber untuk aplikasi Go lengkap yang dapat Anda terapkan ke layanan runtime Go.

Example main.go

```
package main
import (
    "fmt"
    "net/http"
)

func main() {
    http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
        fmt.Fprint(w, "<h1>Welcome to App Runner</h1>")
    })
    fmt.Println("Starting the server on :3000...")
    http.ListenAndServe(":3000", nil)
}
```

Example apprunner.yaml

```
version: 1.0
runtime: go1
build:
  commands:
    build:
      - go build main.go
run:
  command: ./main
  network:
    port: 3000
    env: APP_PORT
```

Informasi rilis Go runtime

Topik ini mencantumkan detail lengkap untuk versi runtime Go yang didukung App Runner.

Versi runtime yang didukung — build App Runner asli

Nama runtime	Versi minor	Paket termasuk
Go 1 (pergi 1)	1.18.10	
	1.18.9	
	1.18.8	
	1.18.7	

> **Note**
>
> App Runner menyediakan proses build yang direvisi untuk runtime utama tertentu yang telah dirilis baru-baru ini. Karena itu, Anda akan melihat referensi ke build App Runner yang direvisi dan build App Runner asli di bagian tertentu dari dokumen ini. Untuk informasi selengkapnya, lihat Versi runtime terkelola dan build App Runner.

Mengembangkan kode aplikasi untuk App Runner

Bab ini membahas informasi runtime dan pedoman pengembangan yang harus Anda pertimbangkan saat mengembangkan atau memigrasi kode aplikasi untuk penerapan. AWS App Runner

Informasi runtime

Baik Anda memberikan image container atau App Runner membuat satu untuk Anda, App Runner menjalankan kode aplikasi Anda dalam instance container. Berikut adalah beberapa aspek kunci dari lingkungan runtime instance container.

- Dukungan Framework — App Runner mendukung gambar apa pun yang mengimplementasikan aplikasi web. Ini agnostik untuk bahasa pemrograman yang Anda pilih dan ke server aplikasi web atau kerangka kerja yang Anda gunakan, jika Anda menggunakannya. Demi kenyamanan Anda, kami menyediakan runtime terkelola khusus platform untuk berbagai platform pemrograman, untuk merampingkan proses pembuatan aplikasi dan pembuatan gambar abstrak.
- Permintaan web — App Runner menyediakan dukungan untuk HTTP 1.0 dan HTTP 1.1 ke instance container. Untuk informasi selengkapnya tentang mengonfigurasi layanan Anda, lihatthe section called “Konfigurasi”. Anda tidak perlu menerapkan penanganan lalu lintas aman HTTPS. App Runner mengalihkan semua permintaan HTTP yang masuk ke titik akhir HTTPS yang sesuai. Anda tidak perlu mengonfigurasi pengaturan apa pun untuk mengaktifkan pengalihan permintaan web HTTP. App Runner menghentikan TLS sebelum meneruskan permintaan ke instance container aplikasi Anda.

> **Note**
>
> - Ada total batas waktu tunggu permintaan 120 detik pada permintaan HTTP. 120 detik termasuk waktu yang dibutuhkan aplikasi untuk membaca permintaan, termasuk isi, dan menyelesaikan penulisan respons HTTP.
> - Batas batas waktu baca dan respons permintaan bergantung pada aplikasi yang Anda gunakan. Aplikasi ini mungkin memiliki batas waktu internal mereka sendiri, seperti server HTTP untuk Python, Gunicorn, memiliki batas batas waktu default 30 detik. Dalam kasus seperti itu, batas waktu tunggu aplikasi mengesampingkan batas waktu 120 detik App Runner.

- Anda tidak perlu mengonfigurasi rangkaian sandi TLS atau parameter lainnya karena App Runner menjadi layanan yang dikelola sepenuhnya, mengelola penghentian TLS untuk Anda.

- Aplikasi stateless — Saat ini App Runner tidak mendukung aplikasi stateful. Oleh karena itu, App Runner tidak menjamin persistensi status di luar durasi pemrosesan satu permintaan web yang masuk.
- Penyimpanan — App Runner secara otomatis menskalakan instans ke atas atau ke bawah untuk aplikasi App Runner Anda sesuai dengan volume lalu lintas yang masuk. Anda dapat mengonfigurasi opsi penskalaan Otomatis untuk aplikasi App Runner Anda. Karena jumlah instans yang saat ini aktif memproses permintaan web didasarkan pada volume lalu lintas yang masuk, App Runner tidak dapat menjamin bahwa file dapat bertahan di luar pemrosesan satu permintaan. Oleh karena itu, App Runner mengimplementasikan sistem file dalam instance container Anda sebagai penyimpanan sementara, yang mensyaratkan bahwa file bersifat sementara. Misalnya, file tidak bertahan saat Anda menjeda dan melanjutkan layanan App Runner.

  App Runner memberi Anda penyimpanan singkat 3 GB dan menggunakan bagian dari penyimpanan singkat 3 GB untuk gambar kontainer yang ditarik, dikompresi, dan tidak terkompresi pada instance. Penyimpanan sementara yang tersisa dapat digunakan oleh layanan App Runner Anda. Namun, ini bukan penyimpanan permanen karena sifatnya yang tanpa kewarganegaraan.

  > **Note**
  >
  > Mungkin ada skenario ketika file penyimpanan tetap ada di seluruh permintaan. Misalnya, jika permintaan berikutnya mendarat pada instance yang sama, file penyimpanan akan tetap ada. Persistensi file penyimpanan di seluruh permintaan dapat berguna dalam situasi tertentu. Misalnya, saat menangani permintaan, Anda dapat menyimpan file cache yang diunduh aplikasi jika permintaan future mungkin membutuhkannya. Ini mungkin mempercepat penanganan permintaan di masa depan, tetapi tidak dapat menjamin peningkatan kecepatan. Kode Anda seharusnya tidak berasumsi bahwa file yang telah diunduh dalam permintaan sebelumnya masih ada.
  > Untuk jaminan caching menggunakan throughput tinggi, penyimpanan data dalam memori latensi rendah, gunakan layanan seperti Amazon. ElastiCache

- Variabel lingkungan — Secara default, App Runner membuat variabel `PORT` lingkungan tersedia di instance container Anda. Anda dapat mengonfigurasi nilai variabel dengan informasi port, dan menambahkan variabel dan nilai lingkungan khusus. Anda juga dapat mereferensikan data

sensitif yang disimpan di AWS Secrets Manageratau AWS Systems Manager Parameter Store sebagai variabel lingkungan. Untuk informasi selengkapnya tentang membuat variabel lingkungan, lihatReferensi variabel Lingkungan.

- Peran instans - Jika kode aplikasi Anda membuat panggilan ke AWS layanan apa pun, menggunakan layanan APIs atau salah satu layanan AWS SDKs, buat peran instance menggunakan AWS Identity and Access Management (IAM). Kemudian, lampirkan ke layanan App Runner Anda saat Anda membuatnya. Sertakan semua izin tindakan AWS layanan yang diperlukan kode Anda dalam peran instans Anda. Untuk informasi selengkapnya, lihat the section called “Peran contoh”.

Pedoman pengembangan kode

Pertimbangkan panduan ini saat mengembangkan kode untuk aplikasi web App Runner.

- Menambal gambar kontainer - Saat memberikan gambar kontainer, Anda bertanggung jawab untuk memperbarui dan menambal gambar-gambar ini secara teratur. Sementara App Runner mengelola infrastruktur, Anda harus memastikan keamanan dan up-to-date status gambar kontainer yang disediakan. Untuk informasi selengkapnya, lihat AWS App Runner Documentation
- Desain kode stateless - Rancang aplikasi web yang Anda terapkan ke layanan App Runner Anda menjadi tanpa kewarganegaraan. Kode Anda harus berasumsi bahwa tidak ada status yang bertahan di luar durasi pemrosesan satu permintaan web yang masuk.
- Hapus file sementara — Saat Anda membuat file, file tersebut disimpan di sistem file, dan mengambil bagian dari alokasi penyimpanan layanan Anda. Untuk menghindari out-of-storage kesalahan, jangan menyimpan file sementara untuk waktu yang lama. Seimbangkan ukuran penyimpanan dengan kecepatan penanganan permintaan saat membuat keputusan caching file.
- Instans startup — App Runner menyediakan lima menit waktu startup instans. Instans Anda harus mendengarkan permintaan pada port mendengarkan yang dikonfigurasi dan menjadi sehat dalam waktu lima menit setelah startup mereka. Selama waktu startup, instance App Runner dialokasikan CPU virtual (vCPU) berdasarkan konfigurasi vCPU Anda. Untuk informasi selengkapnya tentang konfigurasi vCPU yang tersedia, lihat. the section called “Konfigurasi yang didukung App Runner”

  Setelah instance berhasil dijalankan, instans masuk ke keadaan idle dan menunggu permintaan. Anda membayar berdasarkan durasi startup instans, dengan biaya minimum satu menit per instans mulai. Untuk informasi lebih lanjut tenngenai harga, lihat harga AWS App Runner.

Menggunakan konsol App Runner

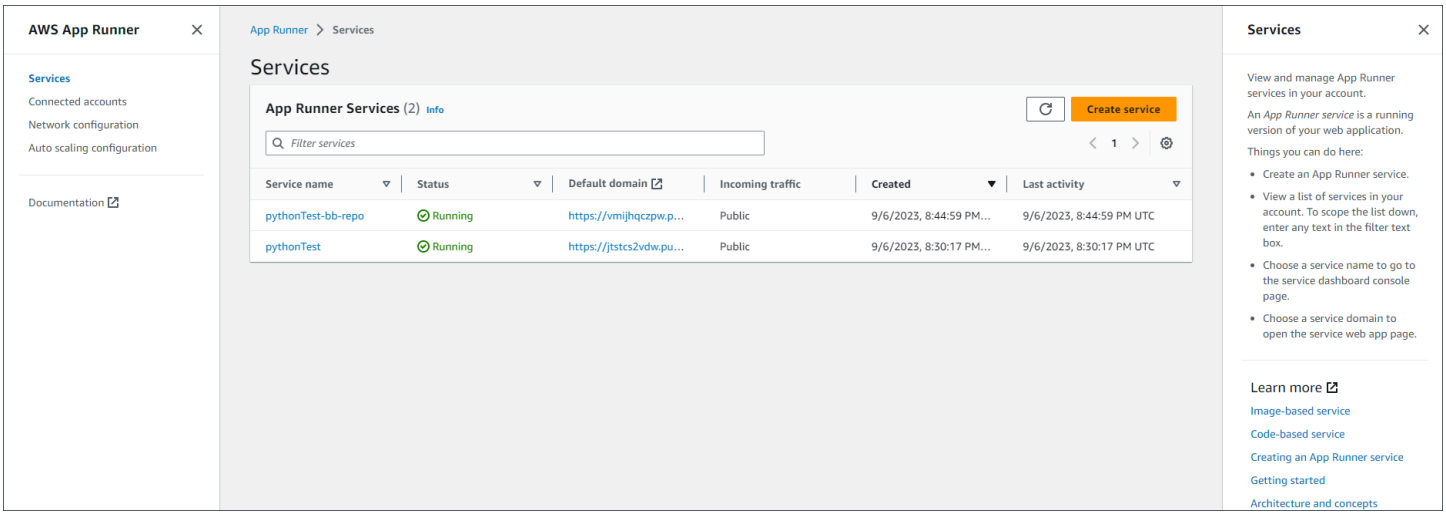
Gunakan AWS App Runner konsol untuk membuat, mengelola, dan memantau layanan App Runner dan sumber daya terkait, seperti akun yang terhubung. Anda dapat melihat layanan yang ada, membuat yang baru, dan mengonfigurasi layanan. Anda dapat melihat status layanan App Runner serta melihat log, memantau aktivitas, dan melacak metrik. Anda juga dapat menavigasi ke situs web layanan Anda atau ke repositori sumber Anda.

Bagian berikut menjelaskan tata letak dan fungsionalitas konsol, dan mengarahkan Anda ke informasi terkait.

Tata letak konsol keseluruhan

Konsol App Runner memiliki tiga area. Dari kiri ke kanan:

- Panel navigasi - Panel samping yang dapat diciutkan atau diperluas. Gunakan untuk memilih halaman konsol tingkat atas yang ingin Anda gunakan.
- Panel konten — Bagian utama dari halaman konsol. Gunakan untuk melihat informasi dan melakukan tugas Anda.
- Panel bantuan — Panel samping untuk informasi lebih lanjut. Perluas untuk mendapatkan bantuan tentang halaman tempat Anda berada. Atau pilih tautan Info apa pun di halaman konsol untuk mendapatkan bantuan kontekstual.



Halaman Layanan

Halaman Layanan mencantumkan layanan App Runner di akun Anda. Anda dapat membuat cakupan daftar dengan menggunakan kotak teks filter.

Untuk sampai ke halaman Layanan

1. Buka konsol App Runner, dan di daftar Wilayah, pilih konsol Anda Wilayah AWS.
2. Pada panel navigasi, silakan pilih Layanan.

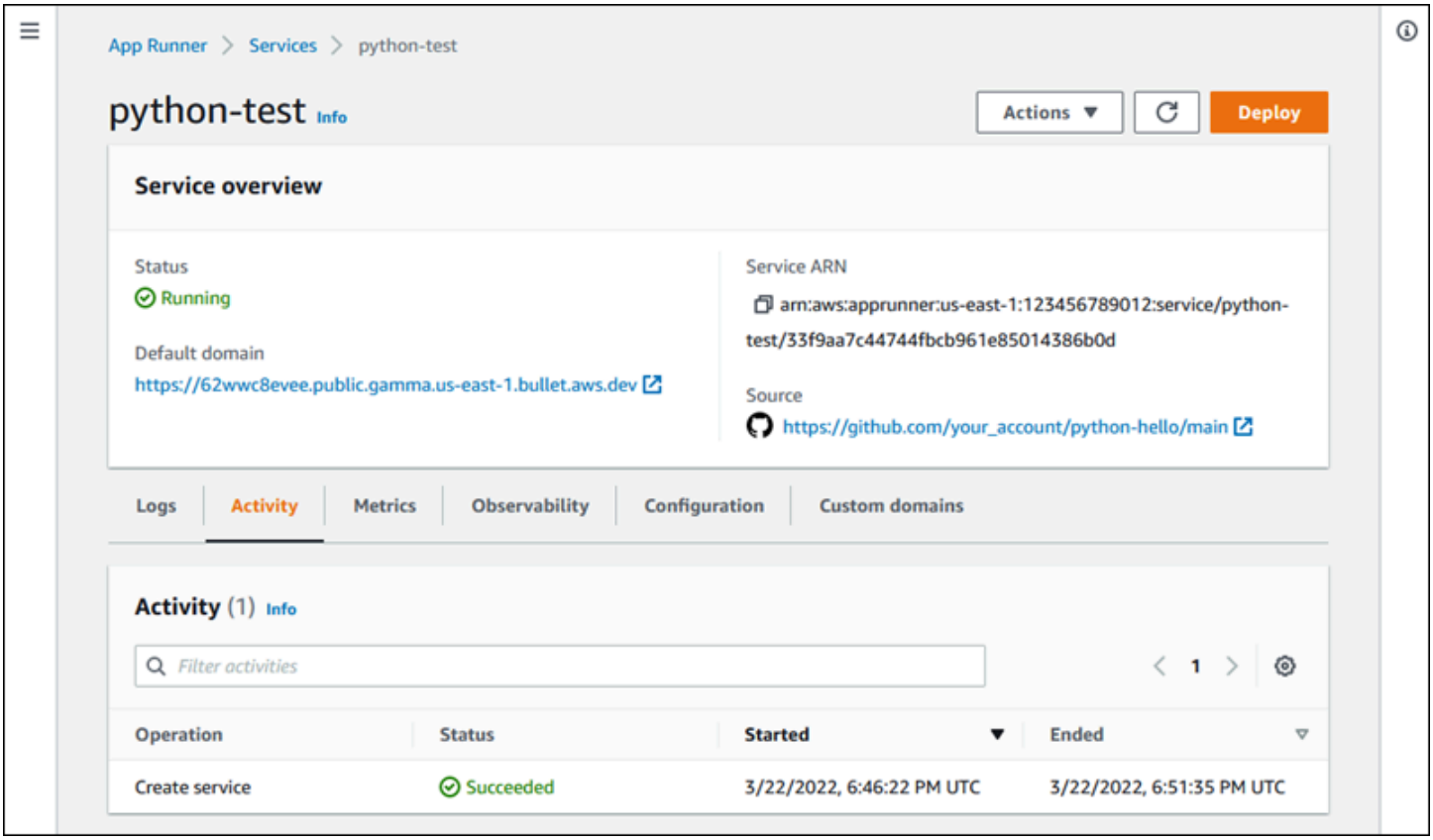
Hal-hal yang dapat Anda lakukan di sini:

- Buat layanan App Runner. Untuk informasi selengkapnya, lihat the section called “Pembuatan”.
- Pilih nama layanan untuk membuka halaman konsol dasbor layanan.
- Pilih domain layanan untuk membuka halaman aplikasi web layanan.

Halaman dasbor layanan

Anda dapat melihat informasi tentang layanan App Runner dan mengelolanya dari halaman dashbaord layanan. Di bagian atas halaman, Anda dapat melihat nama layanan.

Untuk membuka dasbor layanan, navigasikan ke halaman Layanan (lihat bagian sebelumnya), lalu pilih layanan App Runner Anda.



Bagian ikhtisar Layanan memberikan detail dasar tentang layanan App Runner dan aplikasi Anda. Hal-hal yang dapat Anda lakukan di sini:

- Lihat detail layanan seperti status, kesehatan, dan ARN.
- Arahkan ke domain Default —domain yang disediakan App Runner untuk aplikasi web yang berjalan di layanan Anda. Ini adalah subdomain dalam `awsapprunner.com` domain yang dimiliki oleh App Runner.
- Arahkan ke repositori sumber yang digunakan ke layanan.
- Mulai penyebaran repositori sumber ke layanan Anda.
- Jeda, lanjutkan, dan hapus layanan Anda.

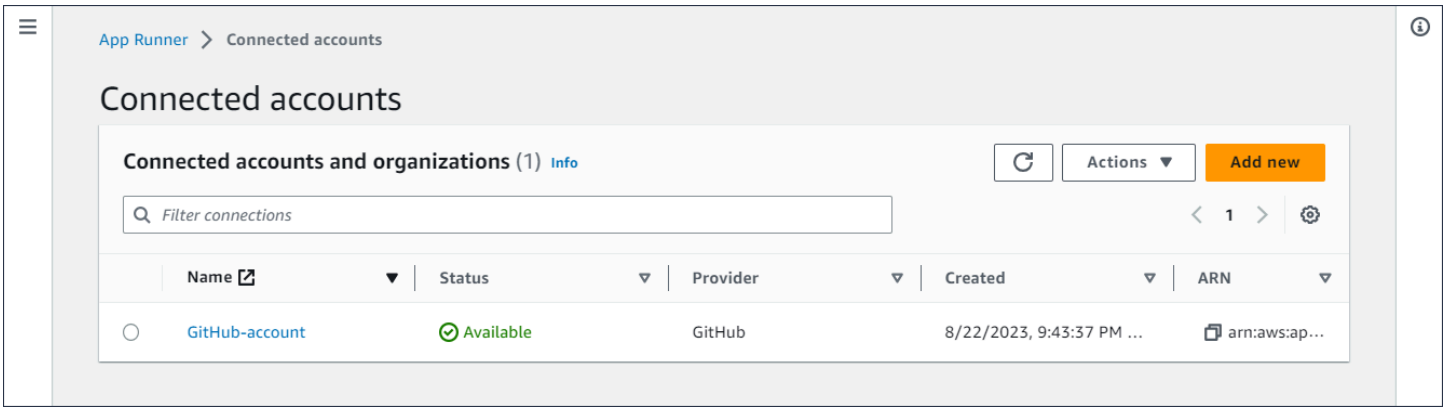
Tab di bawah ikhtisar layanan adalah untuk manajemen layanan dan observabilitas.

Halaman Akun Terhubung

Halaman Akun Terhubung mencantumkan koneksi App Runner ke penyedia repositori kode sumber di akun Anda. Anda dapat membuat cakupan daftar dengan menggunakan kotak teks filter. Untuk informasi selengkapnya tentang akun yang terhubung, lihatthe section called “Koneksi”.

Untuk masuk ke halaman Akun Terhubung

1. Buka konsol App Runner, dan di daftar Wilayah, pilih konsol Anda Wilayah AWS.
2. Di panel navigasi, pilih Akun yang terhubung.



Hal-hal yang dapat Anda lakukan di sini:

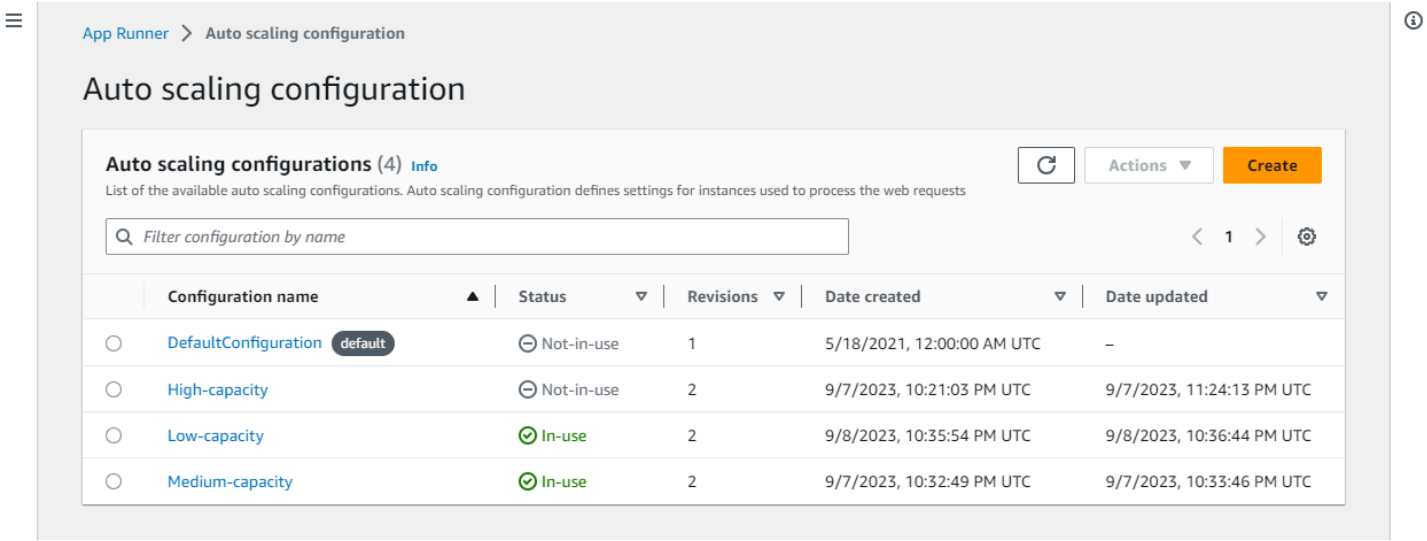
- Lihat daftar koneksi penyedia repositori di akun Anda. Untuk mencakupkan daftar ke bawah, masukkan teks apa pun di kotak teks filter.
- Pilih nama koneksi untuk masuk ke akun atau organisasi penyedia terkait.
- Pilih koneksi untuk menyelesaikan jabat tangan untuk koneksi yang baru saja Anda buat (sebagai bagian dari membuat layanan), atau untuk menghapus koneksi.

Halaman konfigurasi penskalaan Otomatis

Halaman konfigurasi penskalaan otomatis mencantumkan konfigurasi penskalaan otomatis yang telah Anda atur di akun Anda. Anda dapat mengonfigurasi beberapa parameter untuk menyesuaikan perilaku penskalaan otomatis dan menyimpannya dalam konfigurasi berbeda yang nantinya dapat Anda tetapkan ke satu atau beberapa layanan App Runner. Anda dapat membuat cakupan daftar dengan menggunakan kotak teks filter. Untuk informasi selengkapnya tentang konfigurasi penskalaan otomatis, lihat. Mengelola penskalaan otomatis untuk suatu layanan

Untuk masuk ke halaman konfigurasi penskalaan otomatis

1. Buka konsol App Runner, dan di daftar Wilayah, pilih konsol Anda Wilayah AWS.
2. Di panel navigasi, pilih Konfigurasi penskalaan otomatis.



Hal-hal yang dapat Anda lakukan di sini:

- Lihat daftar konfigurasi penskalaan otomatis yang ada di akun Anda.
- Buat konfigurasi penskalaan otomatis baru atau revisi untuk yang sudah ada.
- Tetapkan konfigurasi penskalaan otomatis sebagai default untuk layanan baru yang Anda buat.
- Hapus konfigurasi.
- Pilih nama konfigurasi untuk menavigasi ke panel Revisi penskalaan otomatis untuk mengelola revisi.

Mengelola layanan App Runner

Bab ini menjelaskan cara mengelola AWS App Runner layanan Anda. Dalam Bab ini, Anda mempelajari cara mengelola siklus hidup layanan Anda: membuat, mengonfigurasi, dan menghapus layanan, menerapkan versi aplikasi baru ke layanan Anda, dan mengontrol ketersediaan layanan web Anda dengan menjeda dan melanjutkan layanan Anda. Anda juga mempelajari cara mengelola aspek lain dari layanan Anda, seperti koneksi dan penskalaan otomatis.

Topik

- Membuat layanan App Runner
- Membangun kembali layanan App Runner yang gagal
- Menerapkan versi aplikasi baru ke App Runner
- Mengonfigurasi layanan App Runner
- Mengelola koneksi App Runner
- Mengelola penskalaan otomatis App Runner
- Mengelola nama domain khusus untuk layanan App Runner
- Menjeda dan melanjutkan layanan App Runner
- Menghapus layanan App Runner

Membuat layanan App Runner

AWS App Runner mengotomatiskan transisi dari gambar kontainer atau repositori kode sumber ke layanan web yang sedang berjalan yang menskalakan secara otomatis. Anda mengarahkan App Runner ke gambar atau kode sumber Anda, hanya menentukan sejumlah kecil pengaturan yang diperlukan. App Runner membangun aplikasi Anda jika diperlukan, menyediakan sumber daya komputasi, dan menerapkan aplikasi Anda untuk menjalankannya.

Saat Anda membuat layanan, App Runner membuat sumber daya layanan. Dalam beberapa kasus, Anda mungkin perlu menyediakan sumber daya koneksi. Jika Anda menggunakan konsol App Runner, konsol secara implisit membuat sumber daya koneksi. Untuk informasi selengkapnya tentang jenis sumber daya Pelari Aplikasi, lihatthe section called “Sumber daya App Runner”. Jenis sumber daya ini memiliki kuota yang terkait dengan akun Anda di masing-masing Wilayah AWS. Untuk informasi selengkapnya, lihat the section called “Kuota sumber daya App Runner”.

Ada perbedaan halus dalam prosedur untuk membuat layanan tergantung pada jenis sumber dan penyedia. Topik ini mencakup berbagai prosedur untuk membuat jenis sumber ini sehingga Anda dapat mengikuti mana yang cocok untuk situasi Anda. Untuk memulai prosedur dasar dengan contoh kode, lihatMemulai.

Prasyarat

Sebelum membuat layanan App Runner, pastikan untuk menyelesaikan tindakan berikut:

- Selesaikan langkah-langkah pengaturan diPengaturan.
- Pastikan sumber aplikasi Anda siap. Anda dapat menggunakan repositori kode di GitHub, Bitbucket, atau image container di Amazon Elastic Container Registry (Amazon ECR) Registry (Amazon ECR) untuk membuat layanan App Runner.

Membuat layanan

Bagian ini membahas proses pembuatan untuk dua jenis layanan App Runner: berdasarkan kode sumber, dan berdasarkan gambar kontainer.

> **Note**
>
> Jika Anda membuat konektor VPC lalu lintas keluar untuk suatu layanan, proses startup layanan berikut akan mengalami latensi satu kali. Anda dapat mengatur konfigurasi ini untuk layanan baru saat Anda membuatnya, atau sesudahnya, dengan pembaruan layanan. Untuk informasi selengkapnya, lihat Latensi satu kali di bagian Networking with App Runner dari panduan ini.

Buat layanan dari repositori kode

Bagian berikut menunjukkan cara membuat layanan App Runner ketika sumber Anda adalah repositori kode di GitHubatau Bitbucket. Saat Anda menggunakan repositori kode, App Runner harus terhubung ke organisasi penyedia atau akun. Karena itu, Anda perlu membantu membangun koneksi ini. Untuk informasi selengkapnya tentang koneksi App Runner, lihatthe section called “Koneksi”.

Saat Anda membuat layanan, App Runner akan membuat image Docker yang berisi kode aplikasi dan dependensi Anda. Kemudian meluncurkan layanan yang menjalankan instance kontainer dari gambar ini.

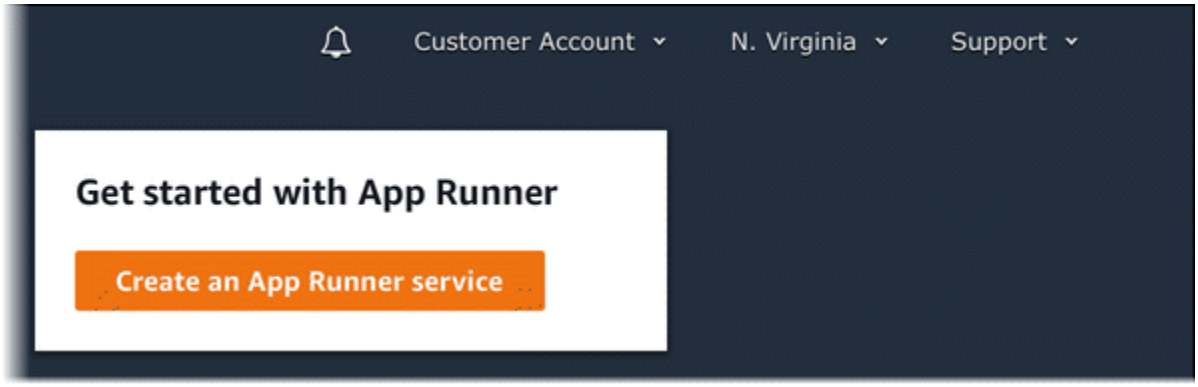
Membuat layanan dari kode menggunakan konsol App Runner

Untuk membuat layanan App Runner menggunakan konsol

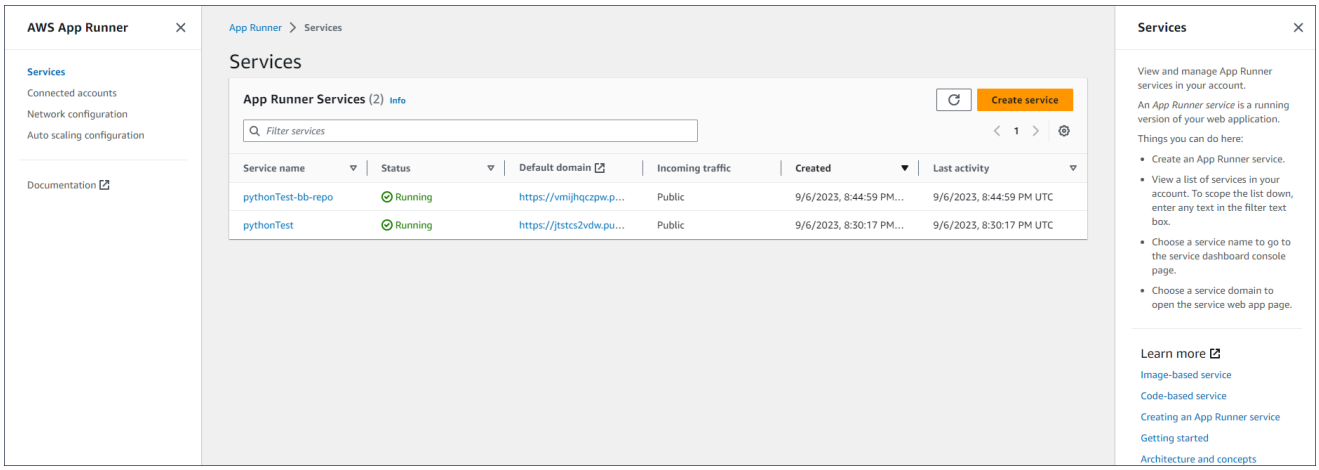
1. Konfigurasikan kode sumber Anda.

   a. Buka konsol App Runner, dan di daftar Wilayah, pilih konsol Anda Wilayah AWS.

   b. Jika Akun AWS belum memiliki layanan App Runner, halaman beranda konsol akan ditampilkan. Pilih Buat layanan App Runner.



   Jika Akun AWS memiliki layanan yang ada, halaman Layanan dengan daftar layanan Anda akan ditampilkan. Pilih Buat layanan.



   c. Pada halaman Sumber dan penyebaran, di bagian Sumber, untuk jenis Repositori, pilih repositori kode sumber.

   d. Pilih Jenis Penyedia. Pilih salah satu GitHubatau Bitbucket.

   e. Selanjutnya pilih akun atau organisasi untuk Penyedia yang telah Anda gunakan sebelumnya, atau pilih Tambah baru. Kemudian, lakukan proses penyediaan kredensi repositori kode Anda dan memilih akun atau organisasi untuk terhubung.

f. Untuk Repositori, pilih repositori yang berisi kode aplikasi Anda.

g. Untuk Branch, pilih cabang yang ingin Anda terapkan.

h. Untuk direktori Source, masukkan direktori di repositori sumber yang menyimpan kode aplikasi dan file konfigurasi Anda.

> **Note**
>
> Perintah build dan start dijalankan dari direktori sumber yang Anda tentukan. App Runner menangani jalur sebagai absolut dari root. Jika Anda tidak menentukan nilai di sini, direktori default ke root repositori.

2. Konfigurasikan penerapan Anda.

   a. Di bagian Pengaturan Deployment, pilih Manual atau Otomatis.

      Untuk informasi selengkapnya tentang metode penerapan, lihatthe section called “Metode deployment”.

   b. Pilih Berikutnya.

Source and deployment Info

Choose the source for your App Runner service and the way it's deployed.

Source and deployment

Source

Repository type

- ○ Container registry
  Deploy your service using a container image stored in a container registry.
- ◉ Source code repository
  Deploy your service using the code hosted in a source repository.

Provider
Choose the provider where you host your code repository.

GitHub

Github Connection Info

App Runner deploys your source code by installing an app called "AWS Connector for GitHub" in your account. You can install this app in your main GitHub account or in a GitHub organization.

myGitHub [Add new]

Repository

python-hello

Branch

main

Source directory
The build and start commands will execute in this directory. App Runner defaults to the root directory if you don't specify a directory here.

/

Leading and trailing slashes ("/") are not required. Valid examples: "apps/targetapp", "/apps/targetapp/", "/targetapp"

Deployment settings

Deployment trigger

- ○ Manual
  Start each deployment yourself using the App Runner console or AWS CLI.
- ◉ Automatic
  Every push to this branch that affects files in the specified **Source directory** deploys a new version of your service.

Cancel Next

3. Konfigurasikan build aplikasi.

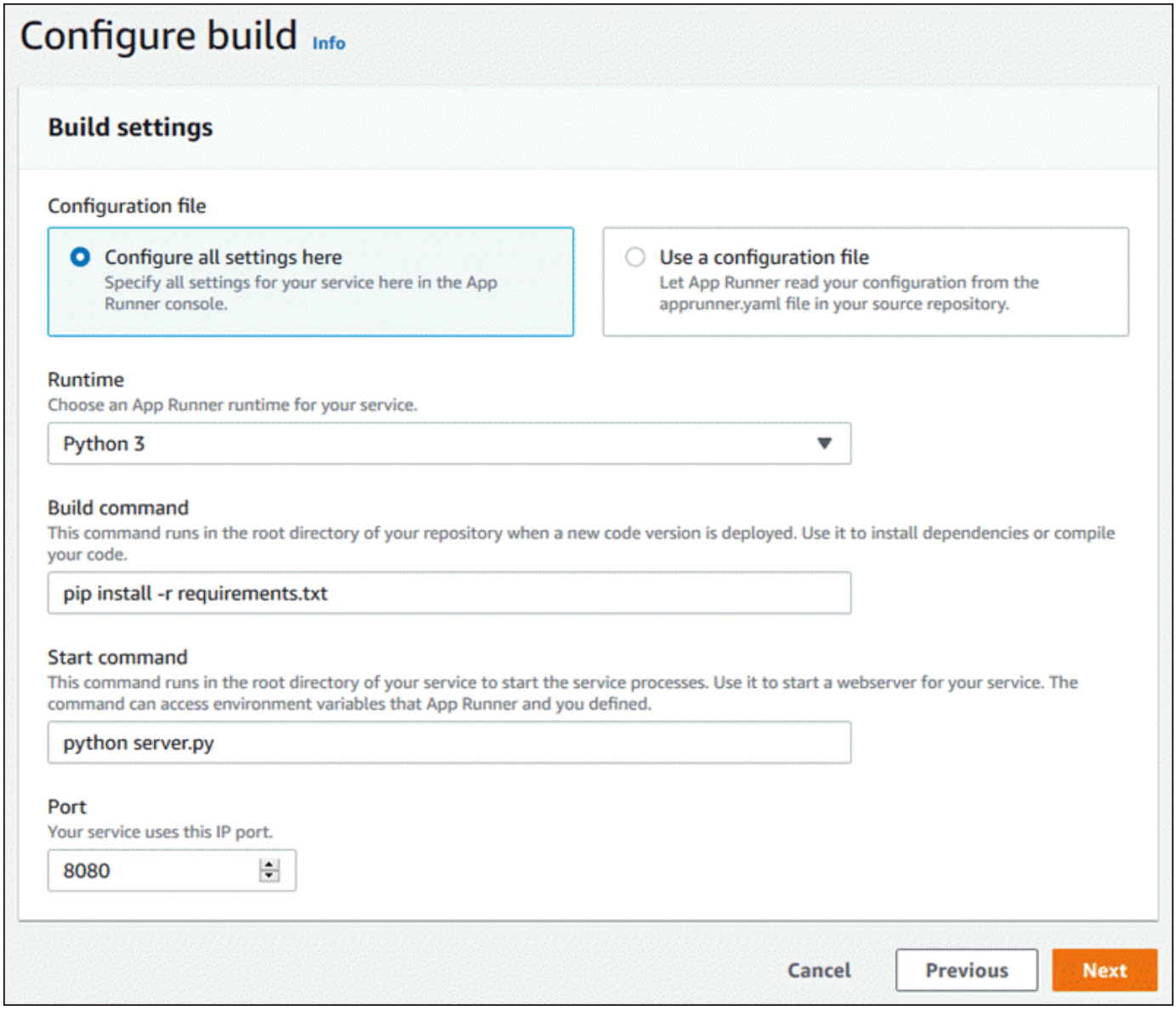
   a. Pada halaman Konfigurasi build, untuk file Konfigurasi, pilih Konfigurasi semua pengaturan di sini jika repositori Anda tidak berisi file konfigurasi App Runner, atau Gunakan file konfigurasi jika ada.

   > **Note**
   >
   > File konfigurasi App Runner adalah cara untuk mempertahankan konfigurasi build Anda sebagai bagian dari sumber aplikasi Anda. Saat Anda menyediakannya, App Runner membaca beberapa nilai dari file dan tidak mengizinkan Anda mengaturnya di konsol.

   b. Berikan setelan build berikut:

      - Runtime — Pilih runtime terkelola tertentu untuk aplikasi Anda.
      - Build command — Masukkan perintah yang membangun aplikasi Anda dari kode sumbernya. Ini mungkin alat khusus bahasa atau skrip yang disediakan dengan kode Anda.
      - Mulai perintah — Masukkan perintah yang memulai layanan web Anda.
      - Port — Masukkan port IP yang didengarkan oleh layanan web Anda.

   c. Pilih Berikutnya.



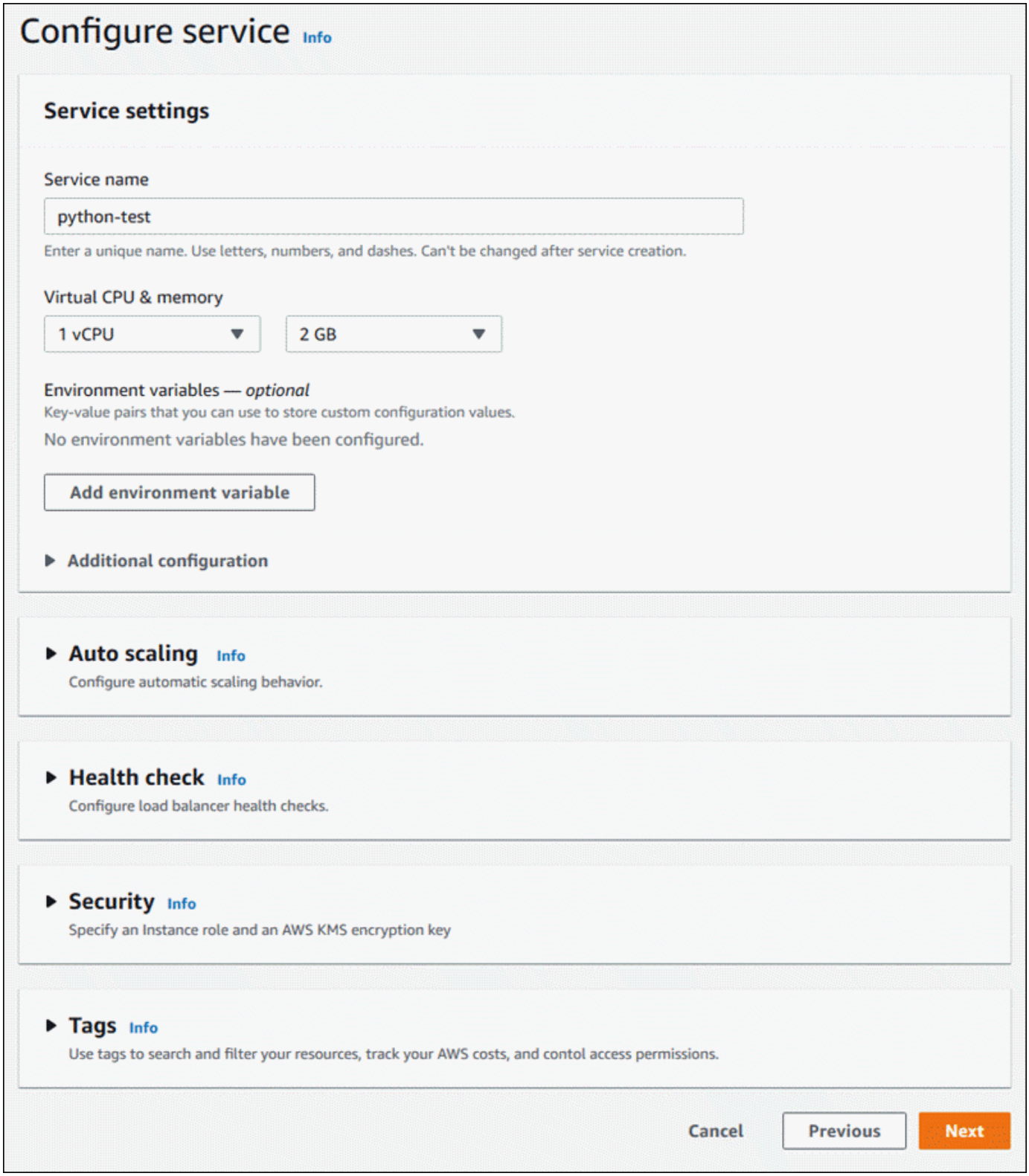
4. Konfigurasikan layanan Anda.

   a. Pada halaman Konfigurasi Layanan, di bagian Pengaturan layanan, masukkan nama layanan.

> **Note**
>
> Semua pengaturan layanan lainnya bersifat opsional atau memiliki default yang disediakan konsol.

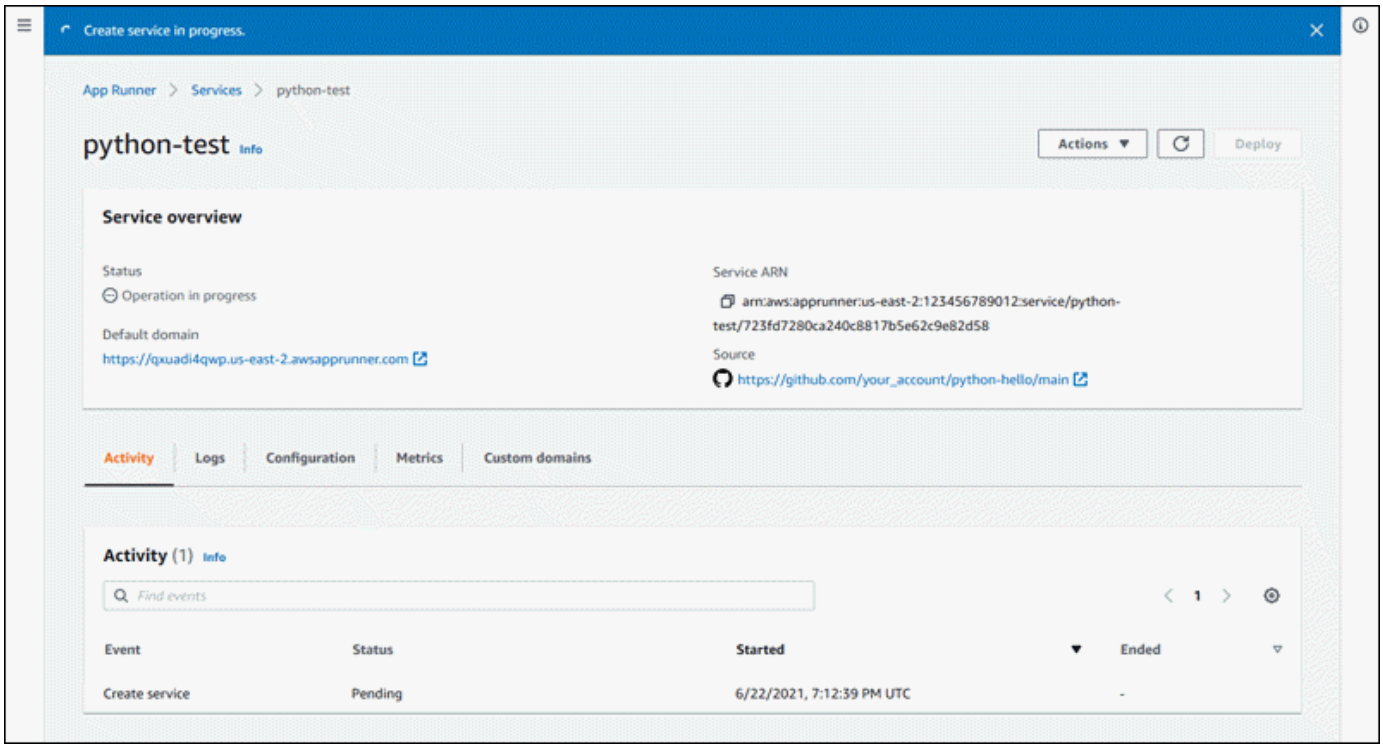
b. Secara opsional mengubah atau menambahkan pengaturan lain untuk memenuhi persyaratan aplikasi Anda.

c. Pilih Berikutnya.



5. Pada halaman Tinjau dan buat, verifikasi semua detail yang Anda masukkan, lalu pilih Buat dan terapkan.

Hasil: Jika layanan berhasil dibuat, konsol menampilkan dasbor layanan dengan ikhtisar Layanan dari layanan baru.



6. Verifikasi bahwa layanan Anda sedang berjalan.

   a. Pada halaman dasbor layanan, tunggu hingga Status layanan Berjalan.

   b. Pilih nilai domain default. Ini adalah URL ke situs web layanan Anda.

   c. Gunakan situs web Anda dan verifikasi bahwa itu berjalan dengan benar.

Membuat layanan dari kode menggunakan App Runner API atau AWS CLI

Untuk membuat layanan menggunakan App Runner API atau AWS CLI, panggil tindakan `CreateService` API. Untuk informasi lebih lanjut dan contoh, lihat CreateService. Jika ini adalah pertama kalinya Anda membuat layanan menggunakan organisasi atau akun tertentu untuk repositori kode sumber (GitHub atau Bitbucket), mulailah dengan menelepon. CreateConnection Ini membuat koneksi antara App Runner dan organisasi atau akun penyedia repositori. Untuk informasi selengkapnya tentang koneksi App Runner, lihatthe section called “Koneksi”.

Jika panggilan mengembalikan respons yang berhasil dengan objek Service yang ditampilkan`"Status": "CREATING"`, layanan Anda akan mulai dibuat.

Untuk contoh panggilan, lihat Membuat layanan repositori kode sumber di Referensi API AWS App Runner

Buat layanan dari gambar Amazon ECR

Bagian berikut menunjukkan cara membuat layanan App Runner saat sumber Anda adalah gambar kontainer yang disimpan di Amazon ECR. Amazon ECR adalah sebuah Layanan AWS. Oleh karena itu, untuk membuat layanan berdasarkan image Amazon ECR, Anda menyediakan App Runner dengan peran akses yang berisi izin tindakan Amazon ECR yang diperlukan.

> **Note**
>
> Gambar yang disimpan di Amazon ECR Public tersedia untuk umum. Jadi, jika gambar Anda disimpan di Amazon ECR Public, peran akses tidak diperlukan.

Saat layanan Anda sedang dibuat, App Runner meluncurkan layanan yang menjalankan instance container dari gambar yang Anda berikan. Tidak ada fase build dalam kasus ini.

Untuk informasi selengkapnya, lihat Layanan berbasis gambar.

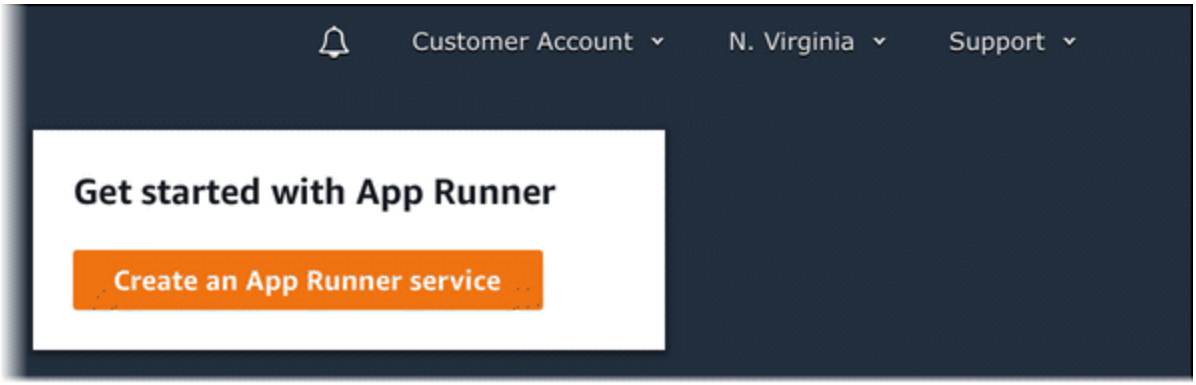
Membuat layanan dari gambar menggunakan konsol App Runner

Untuk membuat layanan App Runner menggunakan konsol

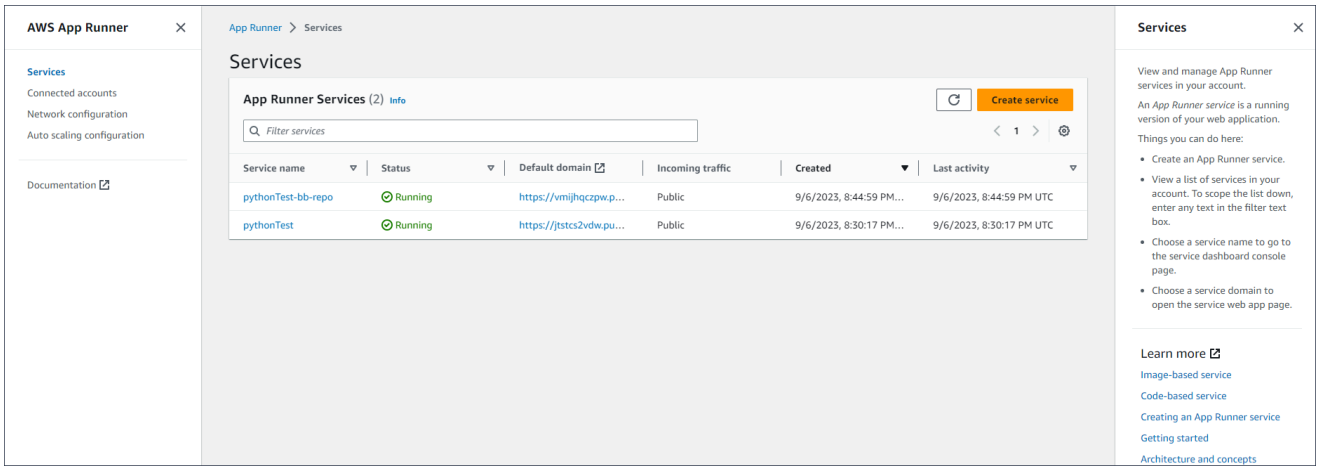
1. Konfigurasikan kode sumber Anda.

   a. Buka konsol App Runner, dan di daftar Wilayah, pilih konsol Anda Wilayah AWS.

   b. Jika Akun AWS belum memiliki layanan App Runner, halaman beranda konsol akan ditampilkan. Pilih Buat layanan App Runner.



Jika Akun AWS memiliki layanan yang ada, halaman Layanan dengan daftar layanan Anda akan ditampilkan. Pilih Buat layanan.



c. Pada halaman Sumber dan penyebaran, di bagian Sumber, untuk jenis Repositori, pilih Registri kontainer.

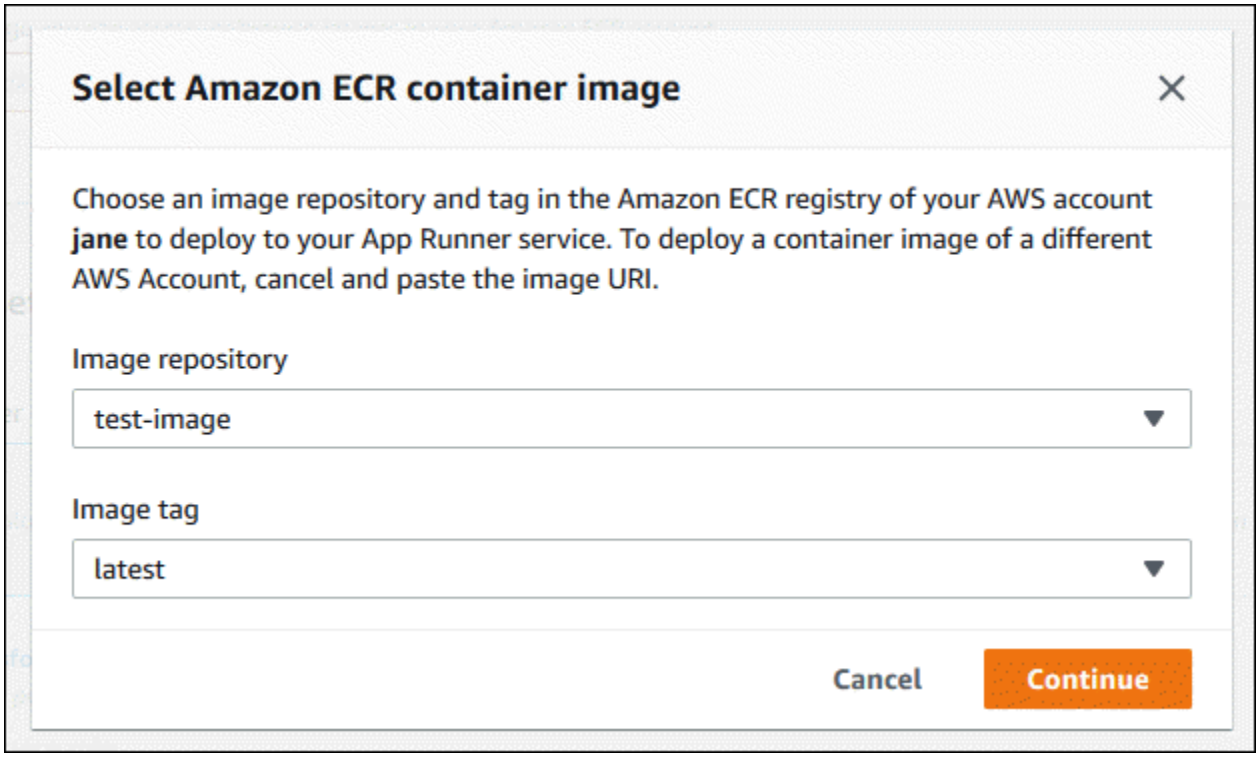
d. Untuk Penyedia, pilih penyedia tempat gambar Anda disimpan:

- Amazon ECR — Gambar pribadi yang disimpan di Amazon ECR.
- Amazon ECR Public - Gambar yang dapat dibaca publik yang disimpan di Amazon ECR Public.

e. Untuk URI gambar Container, pilih Browse.

f. Di kotak dialog Select Amazon ECR container image, untuk Image repository, pilih repositori yang berisi gambar Anda.

g. Untuk tag Gambar, pilih tag gambar tertentu yang ingin Anda gunakan (misalnya, terbaru), lalu pilih Lanjutkan.



2. Konfigurasikan penerapan Anda.

   a. Di bagian Pengaturan Deployment, pilih Manual atau Otomatis.

   > **Note**
   >
   > App Runner tidak mendukung penerapan otomatis untuk gambar Publik Amazon ECR, dan untuk gambar di repositori Amazon ECR yang termasuk dalam AWS akun yang berbeda dari yang digunakan layanan Anda.

   Untuk informasi selengkapnya tentang metode penerapan, lihatthe section called “Metode deployment”.

   b. [Penyedia Amazon ECR] Untuk peran akses ECR, pilih peran layanan yang ada di akun Anda atau pilih untuk membuat peran baru. Jika Anda menggunakan penerapan manual, Anda juga dapat memilih untuk menggunakan peran pengguna IAM pada saat penerapan.

   c. Pilih Berikutnya.

Source and deployment Info

Choose the source for your App Runner service and the way it's deployed.

Source

Repository type

- (●) Container registry
  Deploy your service from a container image stored in a container registry.
- ( ) Source code repository
  Deploy your service from code hosted in a source code repository.

Provider

- (●) Amazon ECR
- ( ) Amazon ECR Public

Container image URI

Enter a URI to an image you can access, or browse images in your Amazon ECR account.

123456789012.dkr.ecr.us-east-2.amazonaws.com/test-image:latest [Browse]

Deployment settings

Deployment trigger

- ( ) Manual
  Start each deployment yourself using the App Runner console or AWS CLI.
- (●) Automatic
  App Runner monitors your registry and deploys a new version of your service for each image push.

ECR access role Info

This role gives App Runner permission to access ECR. To create a custom role, go to the IAM console

- (●) Create new service role
- ( ) Use existing service role

Service role name

The name of an IAM role that App Runner creates in your account with an attached managed policy for ECR access.

AppRunnerECRAccessRole

[Cancel] [Next]

3. Konfigurasikan layanan Anda.

   a. Pada halaman Konfigurasi layanan, di bagian Pengaturan layanan, masukkan nama layanan dan port IP yang didengarkan situs web layanan Anda.

   > **Note**
   >
   > Semua pengaturan layanan lainnya bersifat opsional atau memiliki default yang disediakan konsol.

   b. (Opsional) Ubah atau tambahkan pengaturan lain agar sesuai dengan kebutuhan aplikasi Anda.

   c. Pilih Berikutnya.

Configure service Info

Service settings

Service name

image-test

Enter a unique name. Use letters, numbers, and dashes. Can't be changed after service creation.

Virtual CPU & memory

1 vCPU | 2 GB

Environment variables — *optional*

Key-value pairs that you can use to store custom configuration values.

No environment variables have been configured.

Add environment variable

Port

Your service uses this IP port.

8080

▶ Additional configuration

▶ **Auto scaling** Info

Configure automatic scaling behavior.

▶ **Health check** Info

Configure load balancer health checks.

▶ **Security** Info

Specify an Instance role and an AWS KMS encryption key

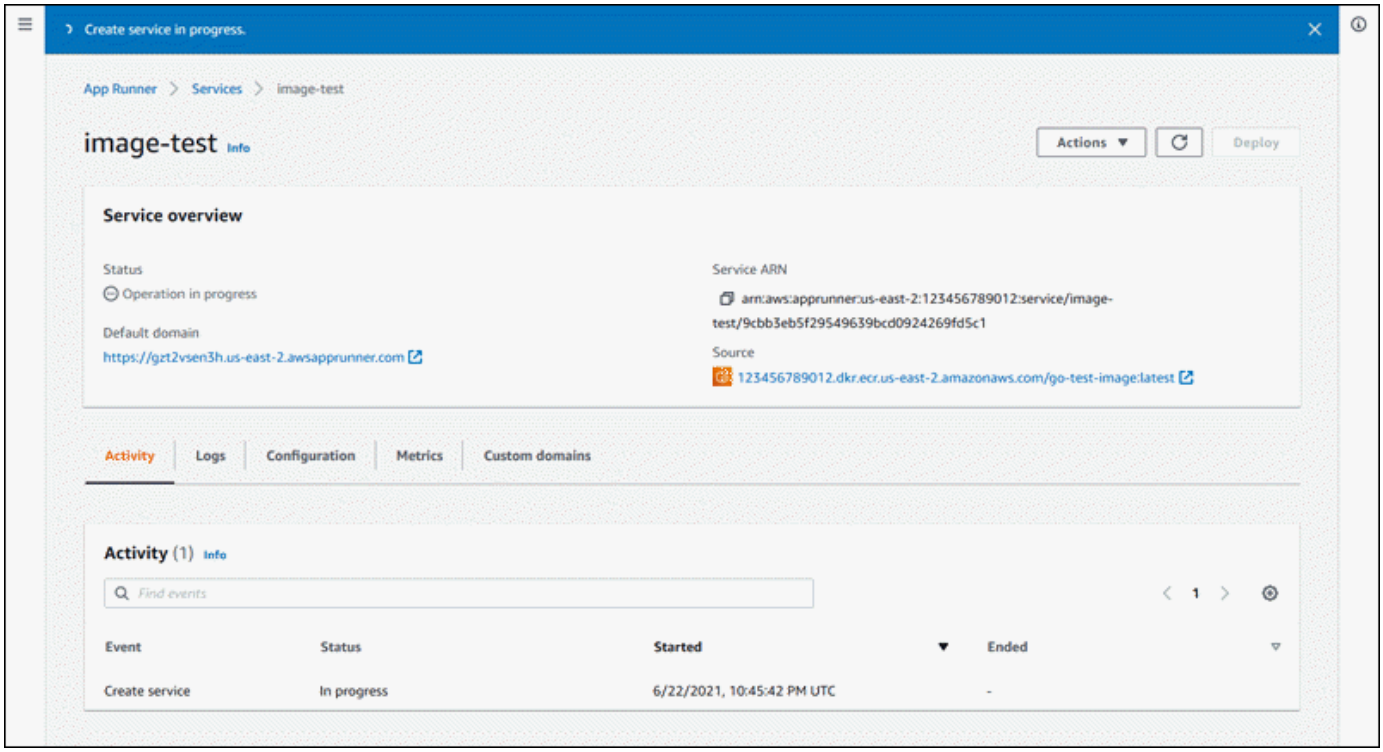
▶ **Tags** Info

Use tags to search and filter your resources, track your AWS costs, and contol access permissions.

Cancel | Previous | Next

4. Pada halaman Tinjau dan buat, verifikasi semua detail yang Anda masukkan, lalu pilih Buat dan terapkan.

   Hasil: Jika layanan berhasil dibuat, konsol menampilkan dasbor layanan, dengan ikhtisar Layanan dari layanan baru.



5. Verifikasi bahwa layanan Anda sedang berjalan.

   a. Pada halaman dasbor layanan, tunggu hingga Status layanan Berjalan.

   b. Pilih nilai domain default. Ini adalah URL ke situs web layanan Anda.

   c. Gunakan situs web Anda dan verifikasi bahwa itu berjalan dengan benar.

Membuat layanan dari gambar menggunakan App Runner API atau AWS CLI

Untuk membuat layanan menggunakan App Runner API atau AWS CLI, panggil tindakan CreateServiceAPI.

Pembuatan layanan Anda dimulai jika panggilan mengembalikan respons yang berhasil dengan objek Service yang ditampilkan`"Status": "CREATING"`.

Untuk contoh panggilan, lihat Membuat layanan repositori gambar sumber di Referensi API AWS App Runner

Membangun kembali layanan App Runner yang gagal

Jika Anda menerima kesalahan Gagal membuat saat membuat layanan App Runner, Anda dapat melakukan salah satu hal berikut.

- Ikuti langkah-langkah the section called “Gagal membuat layanan” untuk mengidentifikasi penyebab kesalahan.
- Jika Anda menemukan kesalahan dalam sumber atau konfigurasi, buat perubahan yang diperlukan dan kemudian bangun kembali layanan Anda.
- Jika masalah sementara dengan App Runner menyebabkan layanan Anda gagal, buat kembali layanan Anda yang gagal tanpa membuat perubahan apa pun pada sumber atau konfigurasi.

Anda dapat membangun kembali layanan yang gagal baik melalui konsol App Runner atau App Runner API atau. AWS CLI

Membangun kembali layanan App Runner yang gagal menggunakan konsol App Runner

Rebuild with updates

Membuat layanan dapat gagal karena berbagai alasan. Ketika ini terjadi, penting untuk mengidentifikasi dan memperbaiki akar penyebab masalah sebelum membangun kembali layanan Anda. Untuk informasi selengkapnya, lihat the section called “Gagal membuat layanan”.

Untuk membangun kembali layanan yang gagal dengan pembaruan

1. Buka tab Konfigurasi di halaman layanan Anda dan pilih Edit.

   Halaman membuka panel ringkasan yang menampilkan daftar semua pembaruan Anda.
2. Buat perubahan yang diperlukan dan tinjau di panel ringkasan.
3. Pilih Simpan dan bangun kembali.

   Anda dapat memantau kemajuan pada tab Log di halaman layanan Anda.

Rebuild without updates

Jika masalah sementara menyebabkan pembuatan layanan Anda gagal, Anda dapat membangun kembali layanan Anda tanpa mengubah pengaturan sumber atau konfigurasinya.

Untuk membangun kembali layanan yang gagal tanpa pembaruan

- Pilih Rebuild di sudut kanan atas halaman layanan Anda.

  Anda dapat memantau kemajuan pada tab Log di halaman layanan Anda.
- Jika layanan Anda gagal dibuat lagi, ikuti petunjuk pemecahan masalah di. the section called "Gagal membuat layanan" Buat perubahan yang diperlukan dan kemudian bangun kembali layanan Anda.

Membangun kembali layanan App Runner yang gagal menggunakan App Runner API atau AWS CLI

Rebuild with updates

Untuk membangun kembali layanan yang gagal:

1. Ikuti instruksi the section called "Gagal membuat layanan" untuk menemukan penyebab kesalahan.
2. Buat perubahan yang diperlukan pada cabang atau gambar repositori sumber atau konfigurasi yang menyebabkan kesalahan.
3. Bangun kembali dengan memanggil tindakan UpdateServiceAPI dengan repositori kode sumber baru atau parameter repositori gambar sumber. App Runner mengambil komit terbaru dari repositori kode sumber.

Example Membangun kembali dengan pembaruan

Dalam contoh berikut, konfigurasi sumber layanan berbasis gambar sedang diperbarui. Nilai `Port` diubah menjadi`80`.

Memperbarui `input.json` file untuk layanan App Runner berbasis gambar

```
{
  "ServiceArn": "arn:aws:apprunner:us-east-1:123456789012:service/python-
app/8fe1e10304f84fd2b0df550fe98a71fa",
  "SourceConfiguration": {
    "ImageRepository": {
      "ImageConfiguration": {
        "Port": "80"
```

```
        }
      }
    }
  }
```

Memanggil tindakan `UpdateService` API.

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

Rebuild without updates

Untuk membangun kembali layanan Anda yang gagal menggunakan App Runner API atau AWS CLI, panggil tindakan UpdateServiceAPI tanpa membuat perubahan apa pun pada sumber atau konfigurasi layanan Anda. Pilih untuk membangun kembali tanpa membuat pembaruan hanya jika pembuatan layanan Anda gagal karena masalah sementara dengan App Runner.

Menerapkan versi aplikasi baru ke App Runner

Saat membuat layanan AWS App Runner, Anda mengonfigurasi sumber aplikasi—gambar kontainer atau repositori sumber. App Runner menyediakan sumber daya untuk menjalankan layanan Anda dan menyebarkan aplikasi Anda ke sana.

Topik ini menjelaskan cara untuk menerapkan kembali sumber aplikasi ke layanan App Runner Anda saat versi baru tersedia. Ini bisa berupa versi gambar baru di repositori gambar atau komit baru di repositori kode. App Runner menyediakan dua metode untuk menerapkan ke layanan: otomatis dan manual.

Metode deployment

App Runner menyediakan metode berikut bagi Anda untuk mengontrol cara penerapan aplikasi dimulai.

Penyebaran otomatis

Gunakan penerapan otomatis saat Anda menginginkan perilaku integrasi dan penerapan berkelanjutan (CI/CD) untuk layanan Anda. App Runner memantau repositori gambar atau kode Anda untuk perubahan.

Repositori gambar - Setiap kali Anda mendorong versi gambar baru ke repositori gambar Anda, atau komit baru ke repositori kode Anda, App Runner secara otomatis menerapkannya ke layanan Anda tanpa tindakan lebih lanjut di pihak Anda.

Repositori kode - Setiap kali Anda mendorong komit baru ke repositori kode Anda yang membuat perubahan di direktori sumber, App Runner menyebarkan seluruh repositori Anda. Karena hanya perubahan dalam direktori sumber yang memicu penerapan otomatis, penting untuk memahami bagaimana lokasi direktori sumber memengaruhi cakupan penerapan otomatis.

- Direktori tingkat atas (root repositori) - Ini adalah nilai default yang ditetapkan untuk direktori sumber saat Anda membuat layanan. Jika direktori sumber Anda diatur ke nilai ini, ini berarti seluruh repositori berada di dalam direktori sumber. Jadi semua komit yang Anda dorong ke repositori sumber akan memicu penerapan dalam kasus ini.
- Jalur direktori apa pun yang bukan root repositori (non-default) - Karena hanya perubahan yang didorong dalam direktori sumber yang akan memicu penerapan otomatis, setiap perubahan yang didorong ke repositori Anda yang tidak ada di direktori sumber tidak akan memicu penerapan otomatis. Oleh karena itu, Anda harus menggunakan penerapan manual untuk menyebarkan perubahan yang Anda dorong di luar direktori sumber.

> **Note**
>
> App Runner tidak mendukung penerapan otomatis untuk gambar Publik Amazon ECR, dan untuk gambar di repositori Amazon ECR yang termasuk dalam AWS akun yang berbeda dari yang digunakan layanan Anda.

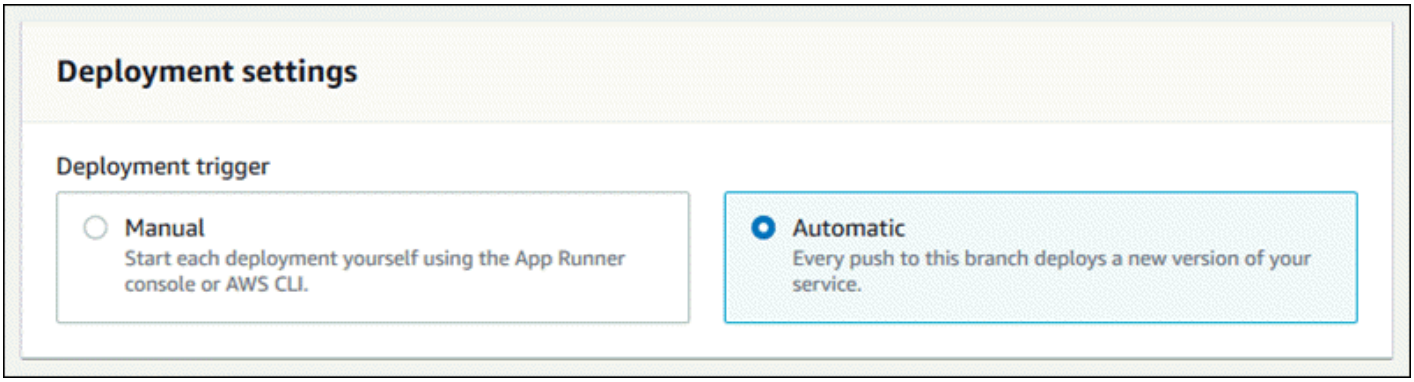
Penyebaran manual

Gunakan penerapan manual saat Anda ingin secara eksplisit memulai setiap penerapan ke layanan Anda. Anda memulai penerapan jika repositori yang dikonfigurasi untuk layanan Anda memiliki versi baru yang ingin Anda terapkan. Untuk informasi selengkapnya, lihat the section called “Penyebaran manual”.

> **Note**
>
> Saat Anda menjalankan penerapan manual, App Runner akan menyebarkan sumber dari repositori lengkap.

Anda dapat mengonfigurasi metode penyebaran untuk layanan Anda dengan cara berikut:

- Konsol — Untuk layanan baru yang Anda buat atau untuk layanan yang sudah ada, di bagian Pengaturan Deployment pada halaman konfigurasi Sumber dan penerapan, pilih Manual atau Otomatis.



- API atau AWS CLI — Dalam panggilan ke UpdateServicetindakan CreateServiceatau, setel `AutoDeploymentsEnabled` anggota SourceConfigurationparameter `False` untuk penerapan manual atau `True` untuk penerapan otomatis.

> **Membandingkan penerapan otomatis dan manual**
>
> Penerapan otomatis dan manual menghasilkan hasil yang sama: kedua metode menerapkan repositori penuh.
> Perbedaan antara kedua metode ini adalah mekanisme pemicunya:
>
> - Penerapan manual dipicu oleh penerapan dari konsol, panggilan ke AWS CLI, atau panggilan ke App Runner API. Penyebaran manualBagian berikut menyediakan prosedur untuk ini.
> - Penerapan otomatis dipicu oleh perubahan dalam isi direktori sumber.

Penyebaran manual

Dengan penerapan manual, Anda perlu secara eksplisit memulai setiap penerapan ke layanan Anda. Jika Anda memiliki versi baru gambar atau kode aplikasi yang siap digunakan, Anda dapat merujuk ke bagian berikut untuk mempelajari cara melakukan penerapan menggunakan konsol dan API.

> **Note**
>
> Saat Anda menjalankan penerapan manual, App Runner akan menyebarkan sumber dari repositori lengkap.

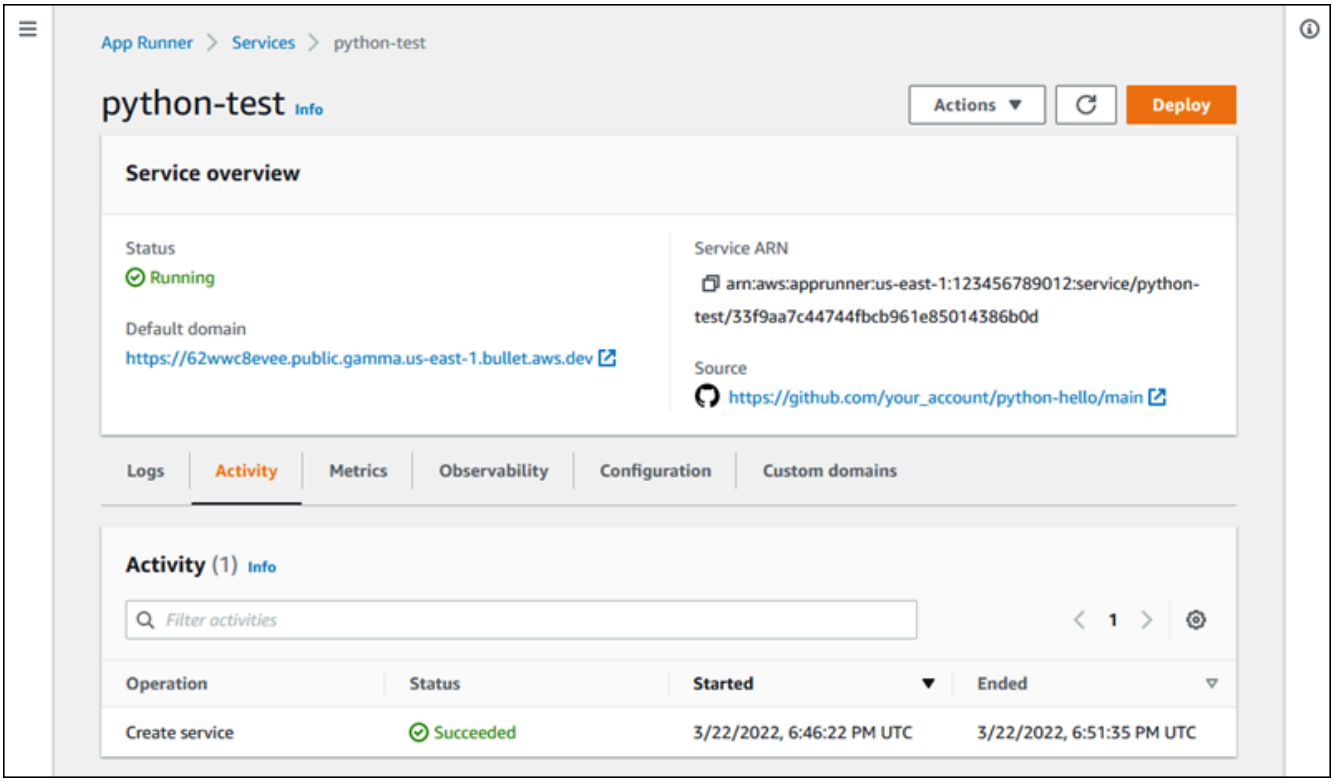
Menerapkan versi aplikasi Anda menggunakan salah satu metode berikut:

App Runner console

Untuk menerapkan menggunakan konsol App Runner

1. Buka konsol App Runner, dan di daftar Wilayah, pilih konsol Anda Wilayah AWS.
2. Di panel navigasi, pilih Layanan, lalu pilih layanan App Runner Anda.

   Konsol menampilkan dasbor layanan dengan ikhtisar Layanan.



3. Pilih Deploy.

   Hasil: Penerapan versi baru dimulai. Pada halaman dasbor layanan, Status layanan berubah menjadi Operasi yang sedang berlangsung.

4. Tunggu hingga penerapan berakhir. Pada halaman dasbor layanan, Status layanan harus berubah kembali ke Running.
5. Untuk memverifikasi bahwa penerapan berhasil, pada halaman dasbor layanan, pilih nilai domain default, yaitu URL ke situs web layanan Anda. Periksa atau berinteraksi dengan aplikasi web Anda dan verifikasi perubahan versi Anda.

> **Note**
>
> Untuk meningkatkan keamanan aplikasi App Runner Anda, domain*.awsapprunner.com terdaftar di Daftar Akhiran Publik (PSL). Untuk keamanan lebih lanjut, kami menyarankan Anda menggunakan cookie dengan `__Host-` awalan jika Anda perlu mengatur cookie sensitif di nama domain default untuk aplikasi App Runner Anda. Praktik ini akan membantu mempertahankan domain Anda dari upaya pemalsuan permintaan lintas situs (CSRF). Untuk informasi selengkapnya, lihat halaman Set-Cookie di Jaringan Pengembang Mozilla.

App Runner API or AWS CLI

Untuk menerapkan menggunakan App Runner API atau AWS CLI, panggil tindakan StartDeploymentAPI. Satu-satunya parameter yang harus dilewati adalah ARN layanan Anda. Anda sudah mengonfigurasi lokasi sumber aplikasi saat membuat layanan, dan App Runner dapat menemukan versi baru. Penerapan Anda dimulai jika panggilan mengembalikan respons yang berhasil.

Mengonfigurasi layanan App Runner

Saat Anda membuat AWS App Runner layanan, Anda menetapkan berbagai nilai konfigurasi. Anda dapat mengubah beberapa pengaturan konfigurasi ini setelah Anda membuat layanan. Pengaturan lain hanya dapat diterapkan saat membuat layanan dan tidak dapat diubah setelahnya. Topik ini membahas konfigurasi layanan Anda menggunakan App Runner API, konsol App Runner, dan file konfigurasi App Runner.

Topik

- Konfigurasikan layanan Anda menggunakan App Runner API atau AWS CLI
- Konfigurasikan layanan Anda menggunakan konsol App Runner

- Konfigurasikan layanan Anda menggunakan file konfigurasi App Runner
- Mengkonfigurasi observabilitas untuk layanan Anda
- Mengkonfigurasi pengaturan layanan menggunakan sumber daya yang dapat dibagikan
- Mengkonfigurasi pemeriksaan kesehatan untuk layanan Anda

Konfigurasikan layanan Anda menggunakan App Runner API atau AWS CLI

API menentukan pengaturan mana yang dapat diubah setelah pembuatan layanan. Daftar berikut membahas tindakan, jenis, dan batasan yang relevan.

- UpdateServicetindakan - Dapat dipanggil setelah pembuatan untuk memperbarui beberapa pengaturan konfigurasi.
  - Dapat diperbarui - Anda dapat memperbarui pengaturan di`SourceConfiguration`,`InstanceConfiguration`, dan `HealthCheckConfiguration` parameter. Namun, di`SourceConfiguration`, Anda tidak dapat mengalihkan jenis sumber Anda dari kode ke gambar atau sebaliknya. Anda harus menyediakan repositoryparameter yang sama seperti yang Anda berikan ketika Anda membuat layanan. Ini salah satu `CodeRepository` atau`ImageRepository`.

    Anda juga dapat memperbarui sumber daya konfigurasi terpisah berikut ARNs yang terkait dengan layanan:

    - `AutoScalingConfigurationArn`
    - `VpcConnectorArn`
  - Tidak dapat diperbarui — Anda tidak dapat mengubah `ServiceName` dan `EncryptionConfiguration` parameter yang tersedia dalam CreateServicetindakan. Mereka tidak dapat diubah setelah dibuat. UpdateServiceTindakan tidak termasuk parameter ini.
  - API vs. File - Anda dapat mengatur `ConfigurationSource` parameter CodeConfigurationtipe (digunakan untuk repositori kode sumber sebagai bagian dari`SourceConfiguration`) ke. `Repository` Dalam hal ini, App Runner mengabaikan pengaturan konfigurasi di`CodeConfigurationValues`, dan membaca pengaturan ini dari file konfigurasi di repositori Anda. Jika Anda menyetel `ConfigurationSource` ke`API`, App Runner mendapatkan semua pengaturan konfigurasi dari panggilan API dan mengabaikan file konfigurasi, meskipun ada.
- TagResourcetindakan - Dapat dipanggil setelah layanan Anda dibuat untuk menambahkan tag ke layanan atau memperbarui nilai tag yang ada.

- UntagResourcetindakan - Dapat dipanggil setelah layanan Anda dibuat untuk menghapus tag dari layanan.

> **Note**
>
> Jika Anda membuat konektor VPC lalu lintas keluar untuk suatu layanan, proses startup layanan berikut akan mengalami latensi satu kali. Anda dapat mengatur konfigurasi ini untuk layanan baru saat Anda membuatnya, atau sesudahnya, dengan pembaruan layanan. Untuk informasi selengkapnya, lihat Latensi satu kali di bagian Networking with App Runner dari panduan ini.

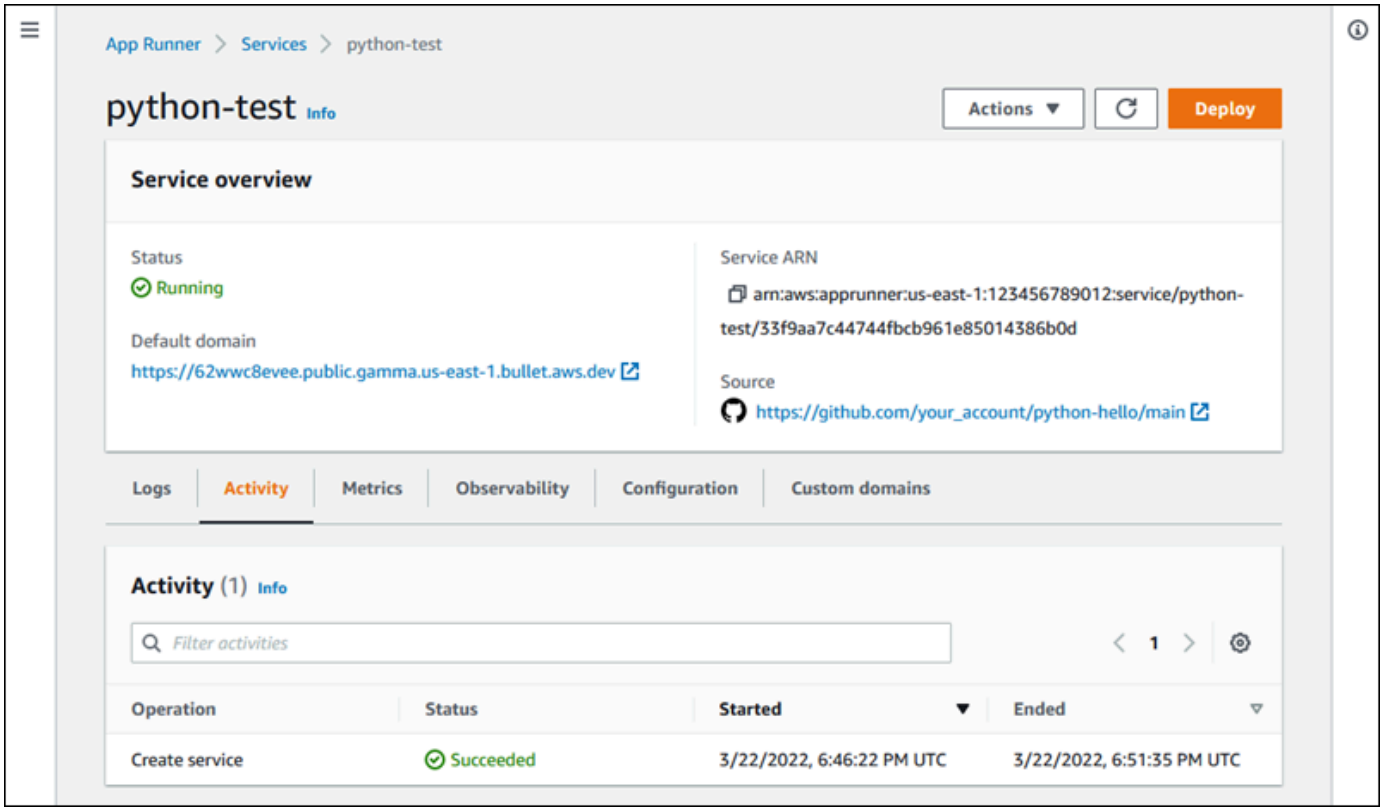
Konfigurasikan layanan Anda menggunakan konsol App Runner

Konsol menggunakan App Runner API untuk menerapkan pembaruan konfigurasi. Aturan pembaruan yang diberlakukan API, seperti yang didefinisikan di bagian sebelumnya, menentukan apa yang dapat Anda konfigurasikan menggunakan konsol. Beberapa pengaturan yang tersedia selama pembuatan layanan tidak tersedia untuk modifikasi nanti. Selain itu, jika Anda memutuskan untuk menggunakan file konfigurasi, pengaturan tambahan disembunyikan di konsol, dan App Runner membacanya dari file.

Untuk mengonfigurasi layanan Anda

1. Buka konsol App Runner, dan di daftar Wilayah, pilih konsol Anda Wilayah AWS.
2. Di panel navigasi, pilih Layanan, lalu pilih layanan App Runner Anda.

   Konsol menampilkan dasbor layanan dengan ikhtisar Layanan.



3. Pada halaman dasbor layanan, pilih tab Konfigurasi.

   Hasil: Konsol menampilkan pengaturan konfigurasi layanan Anda saat ini di beberapa bagian: Sumber dan penerapan, Konfigurasi build, dan Konfigurasi layanan.

4. Untuk memperbarui pengaturan dalam kategori apa pun, pilih Edit.
5. Pada halaman edit konfigurasi, buat perubahan yang diinginkan, lalu pilih Simpan perubahan.

> **Note**
>
> Jika Anda membuat konektor VPC lalu lintas keluar untuk suatu layanan, proses startup layanan berikut akan mengalami latensi satu kali. Anda dapat mengatur konfigurasi ini untuk layanan baru saat Anda membuatnya, atau sesudahnya, dengan pembaruan layanan. Untuk informasi selengkapnya, lihat Latensi satu kali di bagian Networking with App Runner dari panduan ini.

Konfigurasikan layanan Anda menggunakan file konfigurasi App Runner

Saat membuat atau memperbarui layanan App Runner, Anda dapat menginstruksikan App Runner untuk membaca beberapa setelan konfigurasi dari file konfigurasi yang Anda berikan sebagai bagian dari repositori sumber. Dengan melakukan ini, Anda dapat mengelola pengaturan yang terkait dengan kode sumber Anda di bawah kendali sumber, bersama dengan kode itu sendiri. File konfigurasi juga menyediakan pengaturan lanjutan tertentu yang tidak dapat Anda atur menggunakan konsol atau API. Untuk informasi selengkapnya, lihat File konfigurasi App Runner.

> **Note**
>
> Jika Anda membuat konektor VPC lalu lintas keluar untuk suatu layanan, proses startup layanan berikut akan mengalami latensi satu kali. Anda dapat mengatur konfigurasi ini untuk layanan baru saat Anda membuatnya, atau sesudahnya, dengan pembaruan layanan. Untuk informasi selengkapnya, lihat Latensi satu kali di bagian Networking with App Runner dari panduan ini.

Mengkonfigurasi observabilitas untuk layanan Anda

AWS App Runner terintegrasi dengan beberapa AWS layanan untuk memberi Anda rangkaian alat observabilitas ekstensif untuk layanan App Runner Anda. Untuk informasi selengkapnya, lihat Observabilitas.

App Runner mendukung mengaktifkan beberapa fitur observabilitas dan mengonfigurasi perilakunya dengan menggunakan sumber daya yang dapat dibagikan yang disebut. ObservabilityConfiguration Anda dapat menyediakan sumber daya konfigurasi observabilitas saat membuat atau memperbarui layanan. Konsol App Runner membuatnya untuk Anda saat Anda membuat layanan App Runner baru. Menyediakan konfigurasi observabilitas adalah opsional. Jika Anda tidak menyediakannya, App Runner menyediakan konfigurasi observabilitas default.

Anda dapat membagikan konfigurasi observabilitas tunggal di beberapa layanan App Runner untuk memastikan mereka memiliki perilaku observabilitas yang sama. Untuk informasi selengkapnya, lihat the section called “Sumber daya konfigurasi”.

Anda dapat mengonfigurasi fitur observabilitas berikut menggunakan konfigurasi observabilitas:

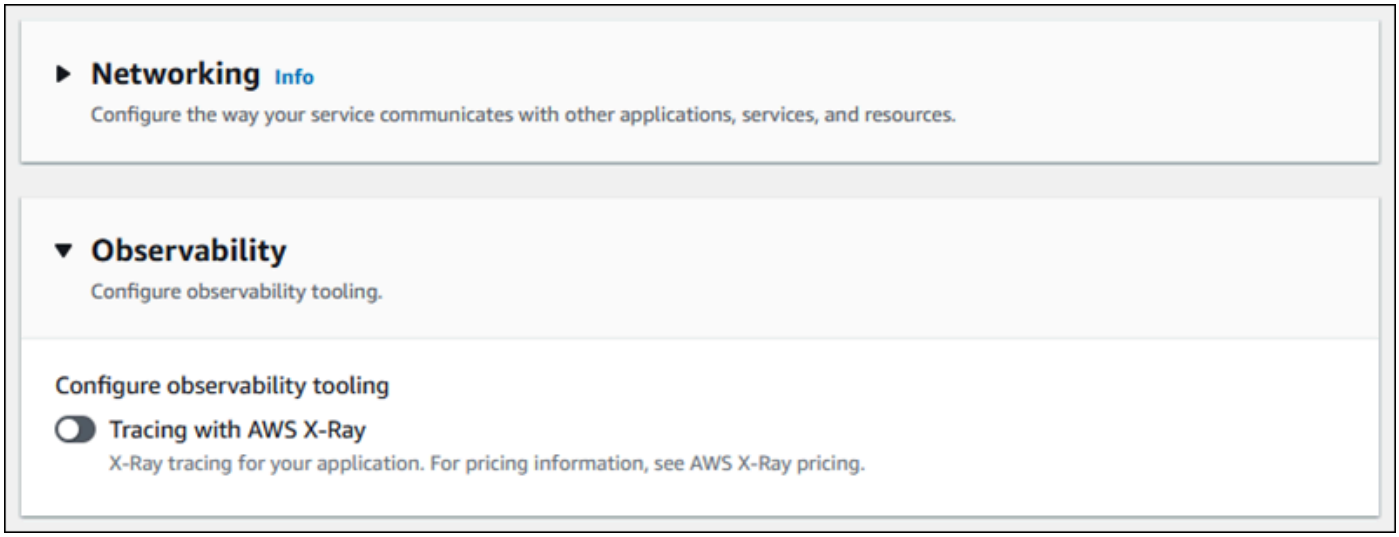
- Konfigurasi pelacakan — Pengaturan untuk melacak permintaan yang disajikan aplikasi Anda dan panggilan hilir yang dibuatnya. Untuk informasi lebih lanjut tentang penelusuran, lihatthe section called “Penelusuran (X-Ray)”.

Kelola observabilitas

Mengelola observabilitas untuk layanan App Runner Anda menggunakan salah satu metode berikut:

App Runner console

Saat membuat layanan menggunakan konsol App Runner, atau saat memperbarui konfigurasinya nanti, Anda dapat mengonfigurasi fitur observabilitas untuk layanan Anda. Cari bagian konfigurasi Observabilitas di halaman konsol.



App Runner API or AWS CLI

Saat memanggil tindakan CreateServiceatau UpdateServiceApp Runner API, Anda dapat menggunakan objek `ObservabilityConfiguration` parameter untuk mengaktifkan fitur observabilitas dan menentukan sumber daya konfigurasi observabilitas untuk layanan Anda.

Gunakan tindakan App Runner API berikut untuk mengelola sumber daya konfigurasi observabilitas Anda.

- CreateObservabilityConfiguration— Membuat konfigurasi observabilitas baru atau revisi ke yang sudah ada.
- ListObservabilityConfigurations— Mengembalikan daftar konfigurasi observabilitas yang terkait dengan Anda Akun AWS, dengan informasi ringkasan.

- DescribeObservabilityConfiguration— Mengembalikan deskripsi lengkap dari konfigurasi observabilitas.
- DeleteObservabilityConfiguration— Menghapus konfigurasi observabilitas. Anda dapat menghapus revisi tertentu atau revisi aktif terbaru. Anda mungkin perlu menghapus konfigurasi observabilitas yang tidak perlu jika Anda mencapai kuota konfigurasi observabilitas untuk Anda. Akun AWS

Mengkonfigurasi pengaturan layanan menggunakan sumber daya yang dapat dibagikan

Untuk beberapa fitur, masuk akal untuk berbagi konfigurasi di seluruh AWS App Runner layanan. Misalnya, Anda mungkin ingin satu set layanan memiliki perilaku penskalaan otomatis yang sama. Atau Anda mungkin menginginkan pengaturan observabilitas yang identik untuk semua layanan Anda. App Runner memungkinkan Anda berbagi setelan dengan menggunakan sumber daya yang dapat dibagikan terpisah. Anda membuat sumber daya yang mendefinisikan setelan konfigurasi untuk fitur, lalu Anda memberikan Nama Sumber Daya Amazon (ARN) sumber daya konfigurasi ini ke satu atau beberapa layanan App Runner.

App Runner mengimplementasikan sumber daya konfigurasi yang dapat dibagikan untuk fitur berikut:

- Penskalaan otomatis
- Observabilitas
- Akses VPC

Halaman dokumen untuk masing-masing fitur ini memberikan informasi tentang pengaturan yang tersedia dan prosedur manajemen.

Fitur yang menggunakan sumber daya konfigurasi terpisah berbagi beberapa ciri dan pertimbangan desain.

- Revisi — Beberapa sumber konfigurasi dapat memiliki revisi. Penskalaan otomatis dan observabilitas adalah contoh dari dua sumber daya konfigurasi yang menggunakan revisi. Dalam kasus ini, setiap konfigurasi memiliki nama dan revisi numerik. Beberapa revisi konfigurasi memiliki nama yang sama dan nomor revisi yang berbeda. Anda dapat menggunakan nama konfigurasi yang berbeda untuk skenario yang berbeda. Untuk setiap nama, Anda dapat menambahkan beberapa revisi untuk menyempurnakan pengaturan untuk skenario tertentu.

Konfigurasi pertama yang Anda buat dengan nama mendapatkan nomor revisi 1. Konfigurasi selanjutnya dengan nama yang sama mendapatkan nomor revisi berturut-turut (dimulai dengan 2). Anda dapat mengaitkan layanan App Runner dengan revisi konfigurasi tertentu atau dengan revisi konfigurasi terbaru.

- Dibagikan - Anda dapat berbagi sumber daya konfigurasi tunggal di beberapa layanan App Runner. Ini berguna jika Anda ingin mempertahankan konfigurasi yang identik di seluruh layanan ini. Khususnya, jika sumber daya Anda mendukung revisi, Anda dapat mengonfigurasi beberapa layanan untuk menggunakan revisi konfigurasi terbaru. Anda dapat melakukannya dengan menentukan hanya nama konfigurasi, tetapi bukan revisi. Layanan apa pun yang Anda konfigurasikan dengan cara ini menerima pembaruan konfigurasi saat Anda memperbarui layanan. Untuk informasi selengkapnya tentang perubahan konfigurasi, lihatthe section called “Konfigurasi”.
- Manajemen sumber daya — Anda dapat menggunakan App Runner untuk membuat dan menghapus konfigurasi. Anda tidak dapat memperbarui konfigurasi secara langsung. Sebagai gantinya, untuk sumber daya yang mendukung revisi, Anda dapat membuat revisi baru ke nama konfigurasi yang ada untuk memperbarui konfigurasi secara efektif.

> **Note**
>
> Untuk penskalaan otomatis, Anda dapat membuat konfigurasi dan beberapa revisi dengan konsol App Runner dan App Runner API. Baik konsol App Runner maupun App Runner API juga dapat menghapus konfigurasi dan revisi. Untuk detail selengkapnya, lihat Mengelola penskalaan otomatis App Runner.
> Untuk jenis konfigurasi lainnya, seperti konfigurasi observabilitas, Anda hanya dapat membuat konfigurasi dengan satu revisi dengan konsol App Runner. Untuk membuat lebih banyak revisi, dan menghapus konfigurasi, Anda harus menggunakan App Runner API.

- Kuota sumber daya — Ada kuota yang ditetapkan untuk jumlah nama konfigurasi unik dan revisi yang dapat Anda miliki untuk sumber daya konfigurasi Anda di masing-masing. Wilayah AWS Jika Anda mencapai kuota ini, Anda harus menghapus nama konfigurasi atau setidaknya beberapa revisinya sebelum Anda dapat membuat lebih banyak. Untuk revisi konfigurasi penskalaan otomatis, Anda dapat menggunakan konsol App Runner atau App Runner API untuk menghapusnya. Untuk detail selengkapnya, lihat Mengelola penskalaan otomatis App Runner. Anda harus menggunakan App Runner API untuk menghapus sumber daya lainnya. Untuk informasi lebih lanjut tentang kuota, lihatthe section called “Kuota sumber daya App Runner”.
- Tidak ada biaya sumber daya - Anda tidak dikenakan biaya tambahan untuk membuat sumber daya konfigurasi. Anda mungkin dikenakan biaya untuk fitur itu sendiri (misalnya, Anda dikenakan

AWS X-Ray biaya normal saat mengaktifkan penelusuran X-Ray), tetapi tidak untuk sumber daya konfigurasi App Runner yang mengonfigurasi fitur untuk layanan Pelari Aplikasi Anda.

Mengkonfigurasi pemeriksaan kesehatan untuk layanan Anda

AWS App Runner memantau kesehatan layanan Anda dengan melakukan pemeriksaan kesehatan. Protokol pemeriksaan kesehatan default adalah TCP. App Runner melakukan ping ke domain yang ditetapkan ke layanan Anda. Anda dapat mengatur protokol pemeriksaan kesehatan ke HTTP. App Runner mengirimkan permintaan HTTP pemeriksaan kesehatan ke aplikasi web Anda.

Anda dapat mengonfigurasi beberapa pengaturan yang terkait dengan pemeriksaan kesehatan. Tabel berikut menjelaskan pengaturan pemeriksaan kesehatan dan nilai defaultnya.

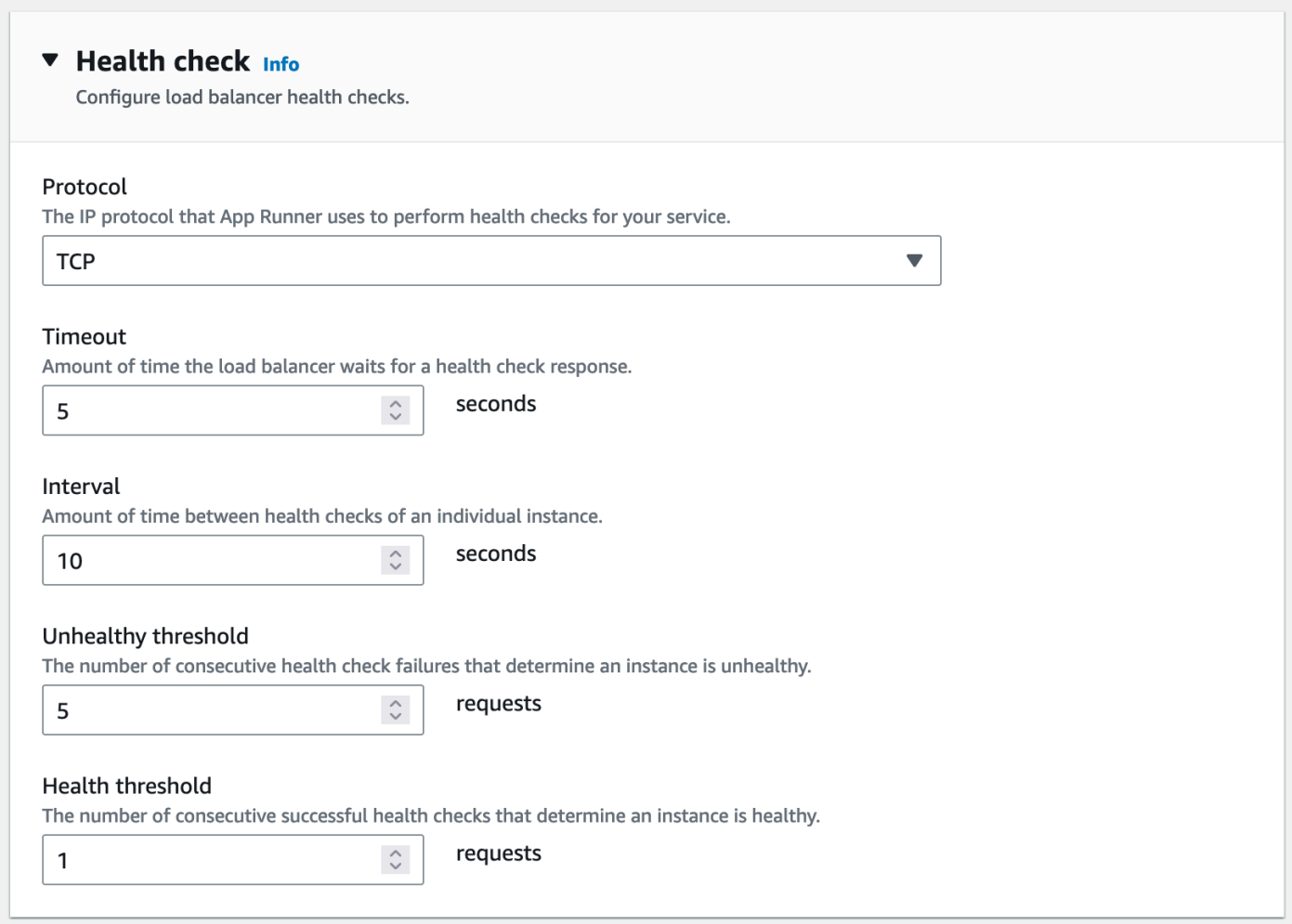
Pengaturan	Deskripsi	Default
Protokol	Protokol IP yang digunakan App Runner untuk melakukan pemeriksaan kesehatan untuk layanan Anda. Jika Anda menyetel protokol`TCP`, App Runner melakukan ping domain default yang ditetapkan ke layanan Anda di port yang didengarkan aplikasi Anda. Jika Anda menyetel protokol`HTTP`, App Runner akan mengirimkan permintaan pemeriksaan kesehatan ke jalur yang dikonfigurasi.	`TCP`
Jalur	URL tempat App Runner mengirimkan permintaan pemeriksaan kesehatan HTTP. Hanya berlaku untuk pemeriksaan HTTP.	`/`
Interval	Interval waktu, dalam hitungan detik, antara pemeriksaan kesehatan.	5
Waktu habis	Waktu, dalam hitungan detik, untuk menunggu respons pemeriksa an kesehatan sebelum memutuskan gagal.	2
Ambang batas yang sehat	Jumlah permiksaan berturut-turut yang harus berhasil sebelum App Runner memutuskan bahwa layanan sehat.	1
Ambang batas yang tidak sehat	Jumlah permiksaan berturut-turut yang harus gagal sebelum App Runner memutuskan bahwa layanan tidak sehat.	5

Konfigurasi pemeriksaan kondisi

Konfigurasikan pemeriksaan kesehatan untuk layanan App Runner Anda menggunakan salah satu metode berikut:

App Runner console

Saat membuat layanan App Runner menggunakan konsol App Runner, atau saat memperbarui konfigurasinya nanti, Anda dapat mengonfigurasi pengaturan pemeriksaan kesehatan. Untuk prosedur konsol lengkap, lihat the section called “Pembuatan” danthe section called “Konfigurasi”. Dalam kedua kasus, cari bagian Konfigurasi pemeriksaan Kesehatan di halaman konsol.



App Runner API or AWS CLI

Saat Anda memanggil tindakan CreateServiceatau UpdateServiceAPI, Anda dapat menggunakan `HealthCheckConfiguration` parameter untuk menentukan pengaturan pemeriksaan kesehatan.

Untuk informasi tentang struktur parameter, lihat HealthCheckConfigurationdi Referensi AWS App Runner API.

Mengelola koneksi App Runner

Saat membuat layanan AWS App Runner, Anda mengonfigurasi sumber aplikasi—gambar kontainer atau repositori sumber yang disimpan dengan penyedia. App Runner harus membuat koneksi yang diautentikasi dan resmi dengan penyedia. Kemudian, App Runner dapat membaca repositori Anda dan menerapkannya ke layanan Anda. App Runner tidak memerlukan pembuatan koneksi saat Anda membuat layanan yang mengakses kode yang disimpan di situs Anda. Akun AWS

App Runner menyimpan informasi koneksi dalam sumber daya yang disebut koneksi. Konsol App Runner dan panduan ini juga merujuk ke koneksi sebagai akun yang terhubung. App Runner memerlukan sumber daya koneksi saat Anda membuat layanan yang memerlukan informasi koneksi pihak ketiga. Berikut ini adalah beberapa informasi penting tentang koneksi:

- Penyedia — App Runner saat ini membutuhkan sumber daya koneksi dengan GitHubatau Bitbucket.
- Bersama — Anda dapat menggunakan sumber daya koneksi untuk membuat beberapa layanan App Runner yang menggunakan akun penyedia repositori yang sama.
- Manajemen sumber daya — Di App Runner, Anda dapat membuat dan menghapus koneksi. Namun, Anda tidak dapat mengubah koneksi yang ada.
- Kuota sumber daya — Sumber daya koneksi memiliki kuota set yang terkait dengan Anda Akun AWS di masing-masing sumber daya. Wilayah AWS Jika Anda mencapai kuota ini, Anda mungkin perlu menghapus sambungan sebelum dapat terhubung ke akun penyedia baru. Anda dapat menghapus sambungan menggunakan konsol App Runner atau API seperti yang dijelaskan di bagian berikut. the section called “Kelola koneksi” Untuk informasi selengkapnya, lihat the section called “Kuota sumber daya App Runner”.

Kelola koneksi

Mengelola koneksi App Runner menggunakan salah satu metode berikut:

App Runner console

Saat Anda menggunakan konsol App Runner untuk membuat layanan, Anda memberikan detail koneksi. Anda tidak perlu secara eksplisit membuat sumber daya koneksi. Di konsol, Anda

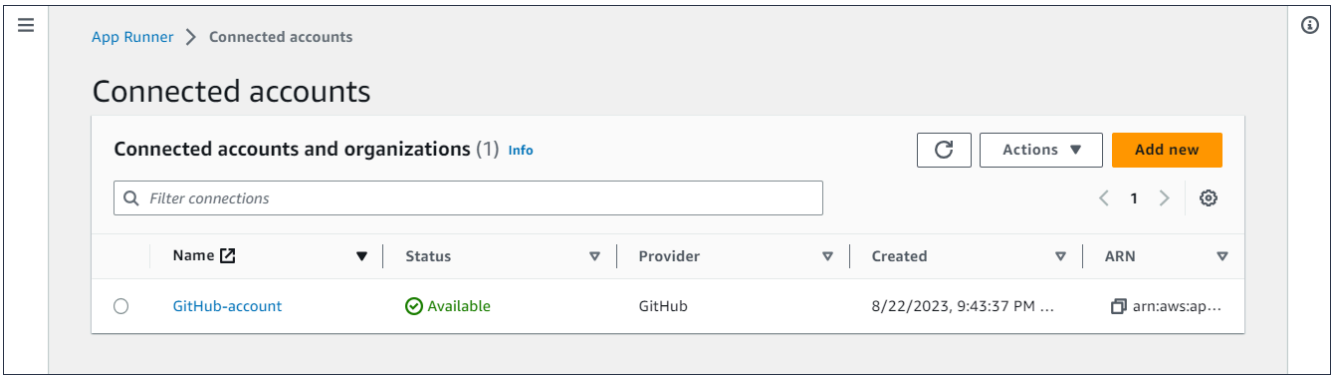
dapat memilih untuk terhubung ke akun GitHub atau Bitbucket yang telah Anda sambungkan sebelumnya, atau terhubung ke akun baru. Bila perlu, App Runner membuat sumber daya koneksi untuk Anda. Untuk koneksi baru, beberapa penyedia meminta Anda untuk menyelesaikan jabat tangan otentikasi sebelum Anda dapat menggunakan koneksi. Konsol membawa Anda melalui proses ini.

Konsol juga memiliki halaman untuk mengelola koneksi yang ada. Anda dapat menyelesaikan jabat tangan otentikasi untuk koneksi jika Anda tidak melakukannya ketika Anda membuat layanan Anda. Anda juga dapat menghapus koneksi yang tidak lagi Anda gunakan. Prosedur berikut menunjukkan bagaimana Anda dapat mengelola koneksi penyedia repositori.

Untuk mengelola koneksi di akun Anda

1. Buka konsol App Runner, dan di daftar Wilayah, pilih konsol Anda Wilayah AWS.
2. Di panel navigasi, pilih Akun yang terhubung.

   Konsol kemudian menampilkan daftar koneksi penyedia repositori di akun Anda.



3. Anda sekarang dapat melakukan salah satu tindakan berikut dengan koneksi apa pun dalam daftar:
   - Buka GitHub/Bitbucket akun atau organisasi — Pilih nama koneksi.
   - Jabat tangan otentikasi lengkap — Pilih koneksi, lalu dari menu Tindakan pilih Jabat tangan lengkap. Konsol membawa Anda melalui proses jabat tangan otentikasi.
   - Hapus koneksi — Pilih koneksi, dan kemudian dari menu Tindakan pilih Hapus. Ikuti instruksi pada prompt penghapusan.

App Runner API or AWS CLI

Anda dapat menggunakan tindakan API App Runner berikut untuk mengelola koneksi Anda.

- CreateConnection— Membuat koneksi ke akun penyedia repositori. Setelah koneksi dibuat, Anda harus menyelesaikan jabat tangan otentikasi secara manual menggunakan konsol App Runner. Proses ini dijelaskan di bagian sebelumnya.
- ListConnections— Mengembalikan daftar koneksi App Runner yang terkait dengan Anda Akun AWS.
- DeleteConnection— Menghapus koneksi. Anda mungkin perlu menghapus koneksi yang tidak perlu jika Anda mencapai kuota koneksi untuk Anda Akun AWS.

Mengelola penskalaan otomatis App Runner

AWS App Runner secara otomatis menskalakan sumber daya komputasi, khususnya instance, naik atau turun untuk aplikasi App Runner Anda. Penskalaan otomatis menyediakan penanganan permintaan yang memadai saat lalu lintas padat, dan mengurangi biaya Anda saat lalu lintas melambat.

Konfigurasi penskalaan otomatis

Anda dapat mengonfigurasi beberapa parameter untuk menyesuaikan perilaku penskalaan otomatis untuk layanan Anda. App Runner mempertahankan pengaturan penskalaan otomatis dalam sumber daya yang dapat dibagikan yang dipanggil. AutoScalingConfiguration Anda dapat membuat dan memelihara konfigurasi penskalaan otomatis yang berdiri sendiri, sebelum Anda menetapkannya ke layanan. Setelah dikaitkan dengan layanan, Anda dapat terus mempertahankan konfigurasi. Anda juga dapat memilih untuk membuat konfigurasi penskalaan otomatis baru saat Anda sedang dalam proses membuat layanan baru atau mengonfigurasi layanan yang sudah ada. Setelah konfigurasi penskalaan otomatis baru dibuat, Anda dapat mengaitkannya ke layanan dan melanjutkan proses pembuatan atau konfigurasi layanan Anda.

Penamaan dan revisi

Konfigurasi penskalaan otomatis memiliki nama dan revisi numerik. Beberapa revisi konfigurasi memiliki nama yang sama dan nomor revisi yang berbeda. Anda dapat menggunakan nama konfigurasi yang berbeda untuk skenario penskalaan otomatis yang berbeda, seperti ketersediaan tinggi atau biaya rendah. Untuk setiap nama, Anda dapat menambahkan beberapa revisi untuk menyempurnakan pengaturan untuk skenario tertentu. Anda dapat memiliki hingga 10 nama konfigurasi penskalaan otomatis unik dan hingga 5 revisi untuk setiap konfigurasi. Jika Anda mencapai batas dan perlu membuat lebih banyak, Anda dapat menghapus satu dan kemudian membuat yang lain. App Runner tidak akan mengizinkan Anda menghapus konfigurasi yang

ditetapkan sebagai default atau digunakan oleh layanan aktif. Untuk informasi lebih lanjut tentang kuota, lihatthe section called “Kuota sumber daya App Runner”.

Mengatur konfigurasi default

Saat membuat atau memperbarui layanan App Runner, Anda dapat menyediakan sumber daya konfigurasi penskalaan otomatis. Menyediakan konfigurasi penskalaan otomatis adalah opsional. Jika Anda tidak menyediakannya, App Runner menyediakan konfigurasi penskalaan otomatis default dengan nilai yang disarankan. Fitur konfigurasi penskalaan otomatis memberi Anda opsi untuk mengatur konfigurasi penskalaan otomatis default Anda sendiri alih-alih menggunakan default yang disediakan App Runner. Setelah Anda menentukan konfigurasi penskalaan otomatis lain sebagai default, konfigurasi tersebut secara otomatis ditetapkan sebagai default ke layanan baru yang Anda buat di masa mendatang. Penunjukan default baru tidak memengaruhi asosiasi yang sebelumnya ditetapkan untuk layanan yang ada.

Mengkonfigurasi layanan dengan penskalaan otomatis

Anda dapat membagikan satu konfigurasi penskalaan otomatis di beberapa layanan App Runner untuk memastikan layanan memiliki perilaku penskalaan otomatis yang sama. Untuk informasi selengkapnya tentang mengonfigurasi konfigurasi penskalaan otomatis dengan konsol App Runner atau App Runner API, lihat bagian yang mengikuti topik ini. Untuk informasi lebih umum tentang sumber daya yang dapat dibagikan, lihatthe section called “Sumber daya konfigurasi”.

Pengaturan yang dapat dikonfigurasi

Anda dapat mengonfigurasi pengaturan penskalaan otomatis berikut:

- Max concurrency — Jumlah maksimum permintaan bersamaan yang diproses instance. Jika jumlah permintaan bersamaan melebihi kuota ini, App Runner meningkatkan skala layanan.
- Ukuran maksimal — Jumlah maksimum instans yang dapat ditingkatkan oleh layanan Anda. Ini adalah jumlah instance tertinggi yang secara bersamaan dapat menangani lalu lintas layanan Anda.
- Ukuran minimum — Jumlah minimum instans yang dapat disediakan oleh App Runner untuk layanan Anda. Layanan selalu memiliki setidaknya jumlah instance yang disediakan ini. Beberapa contoh ini secara aktif menangani lalu lintas. Sisanya adalah bagian dari cadangan kapasitas komputasi hemat biaya, yang siap diaktifkan dengan cepat. Anda membayar untuk penggunaan memori dari semua instance yang disediakan. Anda membayar untuk penggunaan CPU hanya subset aktif.

> **Note**
>
> Jumlah sumber daya vCPU menentukan jumlah instance yang dapat diberikan oleh App Runner ke layanan Anda. Ini adalah nilai kuota yang dapat disesuaikan untuk jumlah sumber daya vCPU Fargate On-Demand yang berada dalam layanan. AWS Fargate Untuk melihat pengaturan kuota vCPU untuk akun Anda atau meminta peningkatan kuota, gunakan konsol Service Quotas di. AWS Management Console Untuk informasi selengkapnya, lihat kuota AWS Fargate layanan di Panduan Pengembang Layanan Kontainer Elastis Amazon.

Mengelola penskalaan otomatis untuk suatu layanan

Kelola penskalaan otomatis untuk layanan App Runner Anda menggunakan salah satu metode berikut:

App Runner console

Saat membuat layanan menggunakan konsol App Runner atau memperbarui konfigurasi layanan, Anda dapat menentukan konfigurasi penskalaan otomatis.

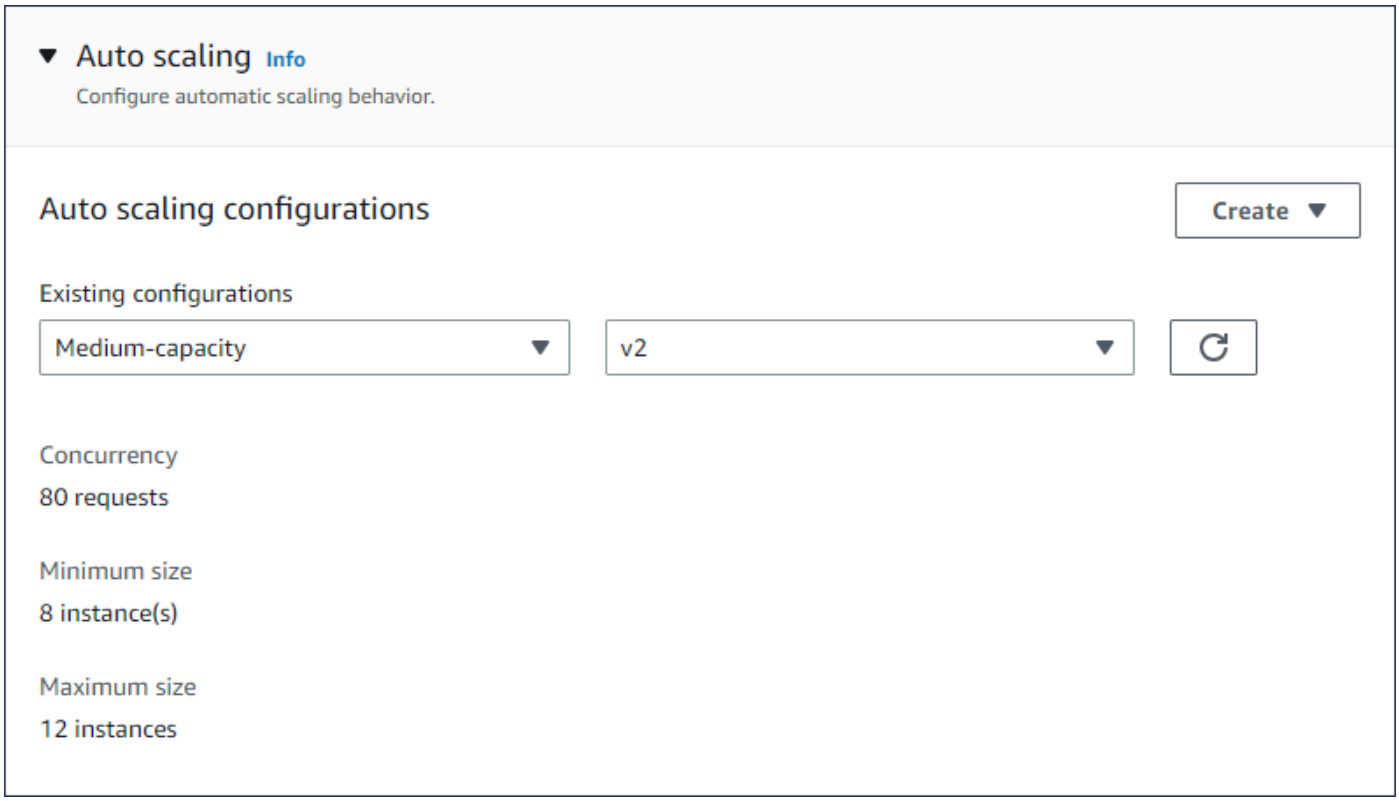
> **Note**
>
> Saat Anda mengubah konfigurasi penskalaan otomatis atau revisi yang terkait dengan layanan, layanan Anda akan di-deploy ulang.

Halaman konfigurasi penskalaan Otomatis menawarkan beberapa opsi untuk mengonfigurasi penskalaan otomatis untuk layanan Anda.

- Untuk menetapkan konfigurasi dan revisi yang ada — Pilih nilai dari drop-down Konfigurasi yang ada. Versi revisi terbaru akan default di drop-down yang berdekatan. Jika ada revisi berbeda yang ingin Anda pilih, lakukan dari drop-down revisi. Nilai konfigurasi untuk tampilan versi revisi.
- Untuk membuat dan menetapkan konfigurasi penskalaan otomatis baru — Pilih Buat ASC baru dari menu Buat. Ini meluncurkan halaman konfigurasi penskalaan otomatis Add custom. Masukkan nama dan nilai Konfigurasi untuk parameter penskalaan otomatis. Kemudian pilih Tambah. App Runner membuat sumber daya konfigurasi penskalaan otomatis baru untuk Anda

dan mengembalikan Anda ke bagian Penskalaan otomatis dengan konfigurasi baru yang dipilih dan ditampilkan.

- Untuk membuat dan menetapkan revisi baru — Pertama pilih nama konfigurasi dari drop-down konfigurasi yang ada. Kemudian pilih Buat revisi ASC dari menu Buat. Ini meluncurkan halaman konfigurasi penskalaan otomatis Add custom. Masukkan nilai untuk parameter penskalaan otomatis. Kemudian pilih Tambah. App Runner membuat revisi konfigurasi penskalaan otomatis baru untuk Anda dan mengembalikan Anda ke bagian Penskalaan otomatis dengan revisi baru yang dipilih dan ditampilkan.



App Runner API or AWS CLI

Saat memanggil tindakan CreateServiceatau UpdateServiceApp Runner API, Anda dapat menggunakan `AutoScalingConfigurationArn` parameter untuk menentukan sumber daya konfigurasi penskalaan otomatis untuk layanan Anda.

Bagian selanjutnya memberikan panduan untuk mengelola sumber daya konfigurasi penskalaan otomatis Anda.

Mengelola sumber daya konfigurasi penskalaan otomatis

Kelola konfigurasi dan revisi penskalaan otomatis App Runner untuk akun Anda menggunakan salah satu metode berikut:

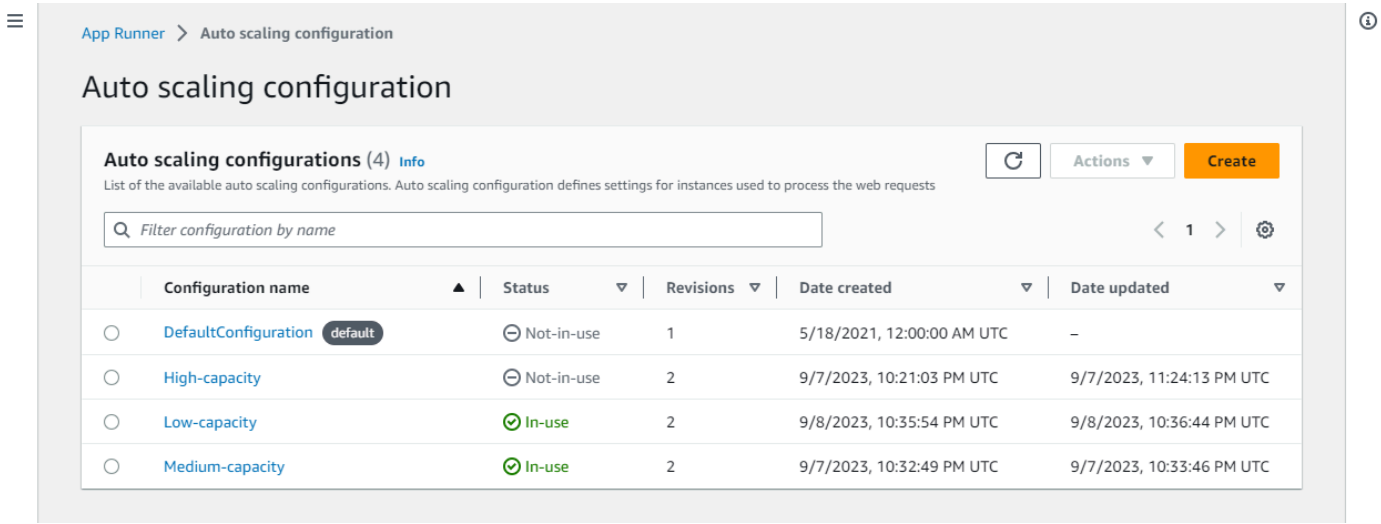
App Runner console

Kelola konfigurasi penskalaan otomatis

Halaman konfigurasi penskalaan otomatis mencantumkan konfigurasi penskalaan otomatis yang telah Anda atur di akun Anda. Anda dapat membuat dan mengelola konfigurasi penskalaan otomatis di halaman ini dan kemudian menetapkannya ke satu atau beberapa layanan App Runner.

Anda dapat melakukan salah satu hal berikut dari halaman ini:

- Buat konfigurasi penskalaan otomatis baru.
- Buat revisi baru untuk konfigurasi penskalaan otomatis yang ada.
- Hapus konfigurasi penskalaan otomatis.
- Tetapkan konfigurasi penskalaan otomatis sebagai default.



Untuk mengelola konfigurasi penskalaan otomatis di akun Anda

1. Buka konsol App Runner, dan di daftar Wilayah, pilih konsol Anda Wilayah AWS.
2. Di panel navigasi, pilih Konfigurasi penskalaan otomatis. Konsol menampilkan daftar konfigurasi penskalaan otomatis di akun Anda.

Anda sekarang dapat melakukan salah satu dari yang berikut ini.

- Untuk membuat konfigurasi penskalaan otomatis baru, ikuti langkah-langkah ini.

  a. Pada halaman Konfigurasi penskalaan otomatis, pilih Buat.

     Halaman konfigurasi Create auto scaling ditampilkan.

  b. Masukkan nilai untuk nama Konfigurasi, Konkurensi, Ukuran minimum, dan Ukuran maksimum.

  c. (Opsional) Jika Anda ingin menambahkan tag, pilih Tag baru otomatis. Kemudian pada bidang yang muncul masukkan Nama dan Nilai (opsional).

  d. Pilih Buat.

- Untuk membuat revisi baru untuk konfigurasi penskalaan otomatis yang ada, ikuti langkah-langkah ini.

  a. Pada halaman Konfigurasi penskalaan otomatis, pilih tombol radio di sebelah konfigurasi yang memerlukan revisi baru. Kemudian pilih Buat revisi dari menu Tindakan.

     Halaman Create revisi ditampilkan.

  b. Aktif, masukkan nilai untuk Konkurensi, Ukuran minimum, dan Ukuran maksimum.

  c. (Opsional) Jika Anda ingin menambahkan tag, pilih Tag baru otomatis. Kemudian pada bidang yang muncul masukkan Nama dan Nilai (opsional).

  d. Pilih Buat.

- Untuk menghapus konfigurasi penskalaan otomatis, ikuti langkah-langkah ini.

  a. Pada halaman Konfigurasi penskalaan otomatis, pilih tombol radio di sebelah konfigurasi yang perlu Anda hapus.

  b. Pilih Hapus dari menu Tindakan.

  c. Untuk melanjutkan penghapusan, pilih Hapus pada dialog konfirmasi. Jika tidak, pilih Batal.

> **Note**
>
> App Runner memvalidasi bahwa pilihan penghapusan Anda tidak ditetapkan sebagai default atau saat ini sedang digunakan oleh layanan aktif apa pun.

- Untuk mengatur konfigurasi penskalaan otomatis sebagai default, ikuti langkah-langkah ini.

  a. Pada halaman Konfigurasi penskalaan otomatis, pilih tombol radio di sebelah konfigurasi yang perlu Anda atur sebagai default.

  b. Pilih Tetapkan sebagai default dari menu Tindakan.

  c. Dialog menampilkan yang memberi tahu Anda bahwa App Runner akan menggunakan revisi terbaru sebagai konfigurasi default untuk semua layanan baru yang Anda buat. Pilih Konfirmasi untuk melanjutkan. Jika tidak, pilih Batal.

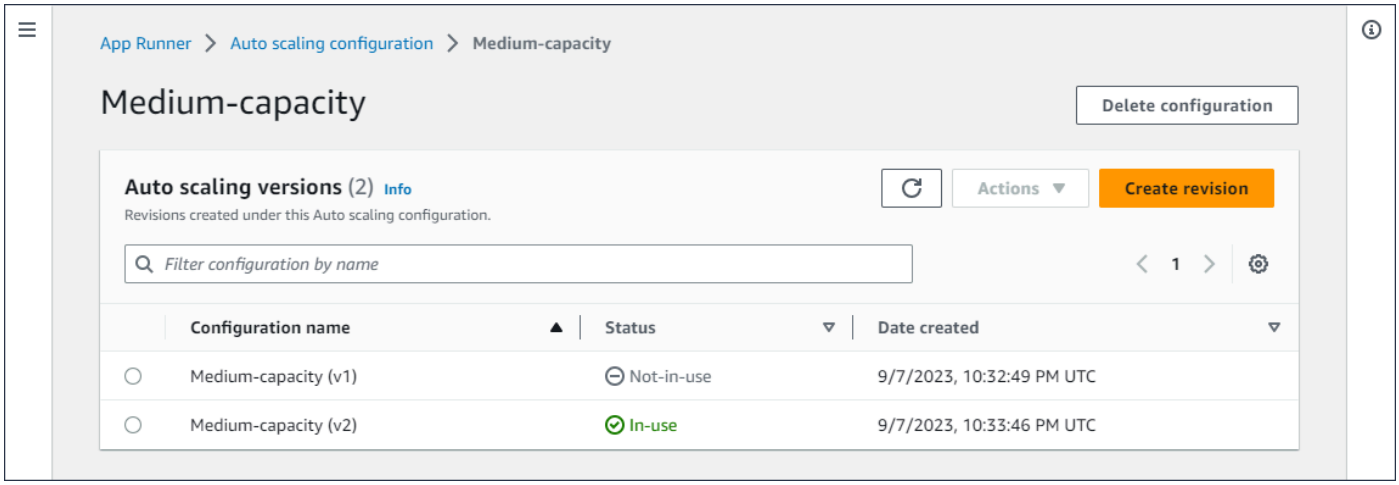
  > **Note**
  >
  > - Saat Anda menetapkan konfigurasi penskalaan otomatis sebagai default, konfigurasi ini secara otomatis ditetapkan sebagai konfigurasi default ke layanan baru yang Anda buat di masa mendatang.
  > - Penunjukan default baru tidak memengaruhi asosiasi yang sebelumnya ditetapkan untuk layanan yang ada.
  > - Jika konfigurasi penskalaan otomatis default yang ditentukan memiliki revisi, App Runner menetapkan revisi terbarunya sebagai default.

Kelola revisi

Konsol ini juga memiliki halaman untuk membuat dan mengelola revisi penskalaan otomatis yang ada yang disebut Revisi penskalaan otomatis. Akses halaman ini dengan memilih nama konfigurasi pada halaman Konfigurasi penskalaan otomatis.

Anda dapat melakukan salah satu hal berikut dari halaman Revisi penskalaan otomatis:

- Buat revisi penskalaan otomatis baru.
- Tetapkan revisi konfigurasi penskalaan otomatis sebagai default.
- Hapus revisi.
- Hapus seluruh konfigurasi penskalaan otomatis, termasuk semua revisi terkait.
- Lihat detail konfigurasi untuk revisi.
- Lihat daftar layanan yang terkait dengan revisi.
- Ubah revisi untuk layanan yang terdaftar.



Untuk mengelola revisi penskalaan otomatis di akun Anda

1. Buka konsol App Runner, dan di daftar Wilayah, pilih konsol Anda Wilayah AWS.
2. Di panel navigasi, pilih Konfigurasi penskalaan otomatis. Konsol menampilkan daftar konfigurasi penskalaan otomatis di akun Anda. Serangkaian prosedur sebelumnya di Kelola konfigurasi penskalaan otomatis bagian ini mencakup gambar layar halaman ini.
3. Sekarang Anda dapat menelusuri konfigurasi penskalaan otomatis tertentu untuk melihat dan mengelola semua revisinya. Di panel Konfigurasi penskalaan otomatis, di bawah kolom Nama konfigurasi, pilih nama konfigurasi penskalaan otomatis. Pilih nama sebenarnya, bukan tombol radio. Ini akan mengarahkan Anda ke daftar semua revisi untuk konfigurasi itu di halaman revisi penskalaan otomatis.
4. Anda sekarang dapat melakukan salah satu dari yang berikut ini.
   - Untuk membuat revisi baru untuk konfigurasi penskalaan otomatis yang ada, ikuti langkah-langkah ini.
     a. Pada halaman Revisi penskalaan otomatis, pilih Buat revisi.

        Halaman Create revisi ditampilkan.

     b. Masukkan nilai untuk Konkurensi, Ukuran minimum, dan Ukuran maksimum.
     c. (Opsional) Jika Anda ingin menambahkan tag, pilih Tag baru otomatis. Kemudian pada bidang yang muncul masukkan Nama dan Nilai (opsional).
     d. Pilih Buat.
   - Untuk menghapus seluruh konfigurasi penskalaan otomatis, termasuk semua revisi terkait, ikuti langkah-langkah berikut.

a. Pilih Hapus konfigurasi di kanan atas halaman.

b. Untuk melanjutkan penghapusan, pilih Hapus pada dialog konfirmasi. Jika tidak, pilih Batal.

> **Note**
>
> App Runner memvalidasi bahwa pilihan penghapusan Anda tidak ditetapkan sebagai default atau saat ini sedang digunakan oleh layanan aktif apa pun.

- Untuk menetapkan revisi penskalaan otomatis sebagai default, ikuti langkah-langkah ini.

  a. Pilih tombol radio di sebelah revisi yang perlu Anda atur sebagai default.

  b. Pilih Tetapkan sebagai default dari menu Tindakan.

  > **Note**
  >
  > - Saat Anda menetapkan konfigurasi penskalaan otomatis sebagai default, konfigurasi ini secara otomatis ditetapkan sebagai konfigurasi default ke layanan baru yang Anda buat di masa mendatang.
  > - Penunjukan default baru tidak memengaruhi asosiasi yang sebelumnya ditetapkan untuk layanan yang ada.

- Untuk melihat detail konfigurasi untuk revisi, ikuti langkah-langkah ini.

  - Pilih tombol radio di sebelah revisi.

    Detail konfigurasi untuk revisi, termasuk ARN, ditampilkan di panel split bawah. Lihat gambar layar di akhir prosedur ini.

- Untuk melihat daftar layanan yang terkait dengan revisi, ikuti langkah-langkah berikut.

  - Pilih tombol radio di sebelah revisi.

    Panel Services, ditampilkan di panel split bawah, di bawah detail konfigurasi revisi. Panel mencantumkan semua layanan yang menggunakan revisi konfigurasi penskalaan otomatis ini. Lihat gambar layar di akhir prosedur ini.

- Untuk mengubah revisi untuk layanan yang terdaftar, ikuti langkah-langkah ini.

a. Pilih tombol radio di sebelah revisi, jika Anda belum melakukannya.

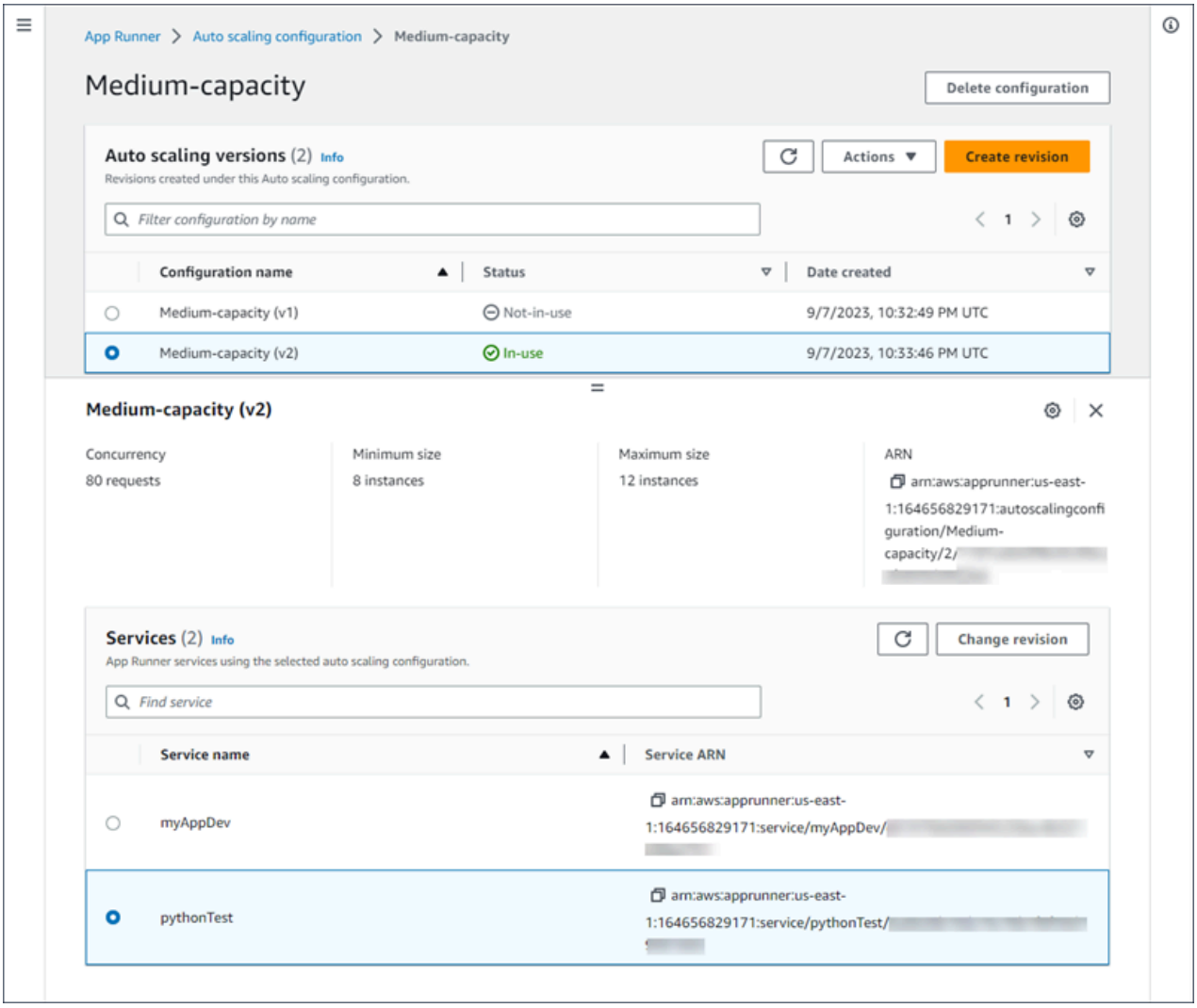
   Panel Services, ditampilkan di panel split bawah, di bawah detail konfigurasi revisi. Panel mencantumkan semua layanan yang menggunakan revisi konfigurasi penskalaan otomatis ini. Lihat gambar layar di akhir prosedur ini.

b. Pada panel Layanan, pilih tombol radio di sebelah layanan yang ingin Anda ubah. Kemudian pilih Ubah revisi.

c. Panel revisi Ubah ASC ditampilkan. Pilih dari revisi yang tersedia di drop-down. Hanya revisi konfigurasi penskalaan otomatis yang Anda pilih sebelumnya yang tersedia. Jika Anda perlu mengubah ke konfigurasi penskalaan otomatis yang berbeda, ikuti prosedur di bawah bagian the section called “Mengelola penskalaan otomatis untuk suatu layanan” sebelumnya.

   Pilih Perbarui untuk melanjutkan perubahan. Jika tidak, pilih Batal.

> **Note**
>
> Saat Anda mengubah revisi yang terkait dengan layanan, layanan Anda akan di-deploy ulang.
> Anda harus memilih refresh pada panel ini untuk melihat asosiasi yang diperbarui.
> Untuk melihat aktivitas yang sedang berlangsung dan status pemindahan layanan, gunakan remah roti panel untuk menavigasi ke App Runner > Services, pilih layanan, lalu lihat tab Log dari panel ikhtisar Layanan.



App Runner API or AWS CLI

Gunakan tindakan App Runner API berikut untuk mengelola sumber daya konfigurasi penskalaan otomatis Anda.

- CreateAutoScalingConfiguration— Membuat konfigurasi penskalaan otomatis baru atau revisi ke yang sudah ada.
- UpdateDefaultAutoScalingConfiguration—Menetapkan konfigurasi penskalaan otomatis menjadi default. Konfigurasi penskalaan otomatis default yang ada akan diatur ke non-default secara otomatis.
- ListAutoScalingConfigurations— Mengembalikan daftar konfigurasi penskalaan otomatis yang terkait dengan Anda Akun AWS, dengan informasi ringkasan.

- ListServicesForAutoScalingConfiguration— Mengembalikan daftar layanan App Runner terkait menggunakan konfigurasi penskalaan otomatis.
- DescribeAutoScalingConfiguration— Mengembalikan deskripsi lengkap tentang konfigurasi penskalaan otomatis.
- DeleteAutoScalingConfiguration— Menghapus konfigurasi penskalaan otomatis. Anda dapat menghapus konfigurasi penskalaan otomatis tingkat atas, revisi tertentu dari satu, atau semua revisi yang terkait dengan konfigurasi tingkat atas. Gunakan `DeleteAllRevisions` parameter opsional untuk menghapus semua revisi. Jika Anda mencapai kuota sumber daya konfigurasi penskalaan otomatis untuk Anda Akun AWS, Anda mungkin perlu menghapus konfigurasi penskalaan otomatis yang tidak perlu.

Mengelola nama domain khusus untuk layanan App Runner

Saat Anda membuat AWS App Runner layanan, App Runner mengalokasikan nama domain untuknya. Ini adalah subdomain di `awsapprunner.com` domain yang dimiliki oleh App Runner. Anda dapat menggunakan nama domain untuk mengakses aplikasi web yang berjalan di layanan Anda.

> **Note**
>
> Untuk meningkatkan keamanan aplikasi App Runner Anda, domain*.awsapprunner.com terdaftar di Daftar Akhiran Publik (PSL). Untuk keamanan lebih lanjut, kami menyarankan Anda menggunakan cookie dengan `__Host-` awalan jika Anda perlu mengatur cookie sensitif di nama domain default untuk aplikasi App Runner Anda. Praktik ini akan membantu mempertahankan domain Anda dari upaya pemalsuan permintaan lintas situs (CSRF). Untuk informasi selengkapnya, lihat halaman Set-Cookie di Jaringan Pengembang Mozilla.

Jika Anda memiliki nama domain, Anda dapat mengaitkannya ke layanan App Runner Anda. Setelah App Runner memvalidasi domain baru Anda, Anda dapat menggunakan domain Anda untuk mengakses aplikasi Anda selain domain App Runner. Anda dapat mengaitkan hingga lima domain kustom.

> **Note**
>
> Anda dapat secara opsional memasukkan `www` subdomain domain Anda. Namun, saat ini hanya didukung oleh API. Konsol App Runner tidak mendukung termasuk `www` subdomain domain Anda.

> **Note**
>
> AWS App Runner tidak mendukung penggunaan zona host pribadi Route 53. Zona yang dihosting pribadi menyesuaikan resolusi nama domain untuk lalu lintas VPC Amazon. Untuk informasi selengkapnya tentang zona yang dihosting pribadi, lihat Bekerja dengan zona yang dihosting pribadi dalam dokumentasi Route 53.

Kaitkan (tautan) domain khusus ke layanan Anda

Saat Anda mengaitkan domain kustom ke layanan Anda, Anda harus menambahkan catatan CNAME dan catatan target DNS ke server DNS Anda. Bagian berikut memberikan informasi tentang catatan CNAME dan catatan target DNS dan cara menggunakannya.

> **Note**
>
> Jika Anda menggunakan Amazon Route 53 sebagai penyedia DNS, App Runner secara otomatis mengonfigurasi domain kustom Anda dengan validasi sertifikat yang diperlukan dan catatan DNS untuk ditautkan ke aplikasi web App Runner Anda. Ini terjadi ketika Anda menggunakan konsol App Runner untuk menautkan domain kustom Anda ke layanan Anda. Mengelola domain kustomTopik berikut memberikan informasi lebih lanjut.

Catatan CNAME

Saat Anda mengaitkan domain kustom dengan layanan Anda, App Runner memberi Anda satu set catatan validasi sertifikat untuk validasi sertifikat. Anda harus menambahkan catatan validasi sertifikat ini ke server Sistem Nama Domain (DNS) Anda. Tambahkan catatan validasi sertifikat, yang disediakan oleh App Runner, ke server DNS Anda. Dengan cara ini, App Runner dapat memvalidasi bahwa Anda memiliki atau mengontrol domain.

> **Note**
>
> Untuk memperbarui sertifikat domain kustom secara otomatis, pastikan Anda tidak menghapus catatan validasi sertifikat dari server DNS Anda. Untuk informasi tentang cara mengatasi masalah yang terkait dengan pembaruan sertifikat, lihatthe section called "Perpanjangan sertifikat domain kustom".

App Runner menggunakan ACM untuk memverifikasi domain. Jika Anda menggunakan catatan CAA dalam catatan DNS Anda, pastikan setidaknya satu referensi catatan CAA. `amazon.com` Jika tidak, ACM tidak dapat memverifikasi domain dan berhasil membuat domain Anda.

Jika Anda menerima kesalahan yang terkait dengan CAA, lihat tautan berikut untuk mempelajari cara mengatasinya:

- Masalah Otorisasi Otoritas Sertifikasi (CAA)
- Bagaimana cara mengatasi kesalahan CAA untuk menerbitkan atau memperbarui sertifikat ACM?
- Nama domain kustom

> **Note**
>
> Jika Anda menggunakan Amazon Route 53 sebagai penyedia DNS, App Runner secara otomatis mengonfigurasi domain kustom Anda dengan validasi sertifikat yang diperlukan dan catatan DNS untuk ditautkan ke aplikasi web App Runner Anda. Ini terjadi ketika Anda menggunakan konsol App Runner untuk menautkan domain kustom Anda ke layanan Anda. Mengelola domain kustomTopik berikut memberikan informasi lebih lanjut.

Catatan target DNS

Tambahkan catatan target DNS ke server DNS Anda untuk menargetkan domain App Runner. Tambahkan satu catatan untuk domain kustom, dan satu lagi untuk `www` subdomain, jika Anda memilih opsi ini. Kemudian, tunggu status domain kustom menjadi Aktif di konsol App Runner. Ini biasanya memakan waktu beberapa menit, tetapi mungkin memakan waktu hingga 24-48 jam (1-2 hari). Ketika domain kustom Anda divalidasi, App Runner mulai merutekan lalu lintas dari domain ini ke aplikasi web Anda.

> **Note**
>
> Untuk kompatibilitas yang lebih baik dengan layanan App Runner, kami sarankan Anda menggunakan Amazon Route 53 sebagai penyedia DNS Anda. Jika Anda tidak menggunakan Amazon Route 53 untuk mengelola catatan DNS publik, hubungi penyedia DNS Anda untuk mengetahui cara menambahkan catatan.
> Jika Anda menggunakan Amazon Route 53 sebagai penyedia DNS, Anda dapat menambahkan CNAME atau alias record untuk subdomain. Untuk domain root, pastikan Anda menggunakan catatan alias.

Anda dapat membeli nama domain dari Amazon Route 53 atau penyedia lain. Untuk membeli nama domain dengan Amazon Route 53, lihat Mendaftarkan domain baru, di Panduan Pengembang Amazon Route 53.

Untuk petunjuk tentang cara mengonfigurasi target DNS di Route 53, lihat Merutekan lalu lintas ke sumber daya Anda, di Panduan Pengembang Amazon Route 53.

Untuk petunjuk tentang cara mengonfigurasi target DNS pada pendaftar lain, seperti, Shopify GoDaddy, Hover, dan sebagainya, lihat dokumentasi spesifik mereka tentang menambahkan catatan Target DNS.

Menentukan domain yang akan diasosiasikan dengan layanan App Runner

Anda dapat menentukan domain yang akan diasosiasikan dengan layanan App Runner dengan cara berikut:

- Domain root — DNS memiliki beberapa batasan inheren yang mungkin menghalangi Anda untuk membuat catatan CNAME untuk nama domain root. Misalnya, jika nama domain Anda`example.com`, Anda dapat membuat catatan CNAME yang merutekan lalu lintas `acme.example.com` ke layanan App Runner Anda. Namun, Anda tidak dapat membuat catatan CNAME yang merutekan lalu lintas `example.com` ke layanan App Runner Anda. Untuk membuat domain root, pastikan Anda menambahkan catatan alias.

  Catatan alias khusus untuk Route 53 dan memiliki keunggulan berikut dibandingkan catatan CNAME:

  - Route 53 memberi Anda lebih banyak fleksibilitas karena catatan alias dapat dibuat untuk domain root atau subdomain. Misalnya, jika nama domain Anda`example.com`, Anda dapat

membuat rekaman yang merutekan permintaan `example.com` atau `acme.example.com` ke layanan App Runner Anda.

- Ini lebih hemat biaya. Ini karena Route 53 tidak mengenakan biaya untuk permintaan yang menggunakan catatan alias untuk merutekan lalu lintas.

- Subdomain — Misalnya, `login.example.com` atau`admin.login.example.com`. Anda juga dapat mengaitkan `www` subdomain sebagai bagian dari operasi yang sama. Anda dapat menambahkan CNAME atau alias record untuk subdomain.
- Wildcard — Misalnya,`*.example.com`. Anda tidak dapat menggunakan `www` opsi dalam kasus ini. Anda dapat menentukan wildcard hanya sebagai subdomain langsung dari domain root dan hanya sendiri. Ini bukan spesifikasi yang valid:`login*.example.com`,`*.login.example.com`. Spesifikasi wildcard ini mengaitkan semua subdomain langsung, dan tidak mengaitkan domain root itu sendiri. Domain root harus dikaitkan dalam operasi terpisah.

Asosiasi domain yang lebih spesifik mengesampingkan yang kurang spesifik. Misalnya, `login.example.com` penggantian`*.example.com`. Sertifikat dan CNAME dari asosiasi yang lebih spesifik digunakan.

Contoh berikut menunjukkan bagaimana Anda dapat menggunakan beberapa asosiasi domain kustom:

1. Kaitkan `example.com` dengan halaman beranda layanan Anda. Aktifkan `www` untuk mengasosiasikan`www.example.com`.
2. Kaitkan `login.example.com` dengan halaman login layanan Anda.
3. Kaitkan `*.example.com` dengan halaman kustom “tidak ditemukan”.

Putuskan (putuskan tautan) domain khusus

Anda dapat memisahkan (memutuskan tautan) domain kustom dari layanan App Runner Anda. Saat Anda memutuskan tautan domain, App Runner berhenti merutekan lalu lintas dari domain ini ke aplikasi web Anda.

> **Note**
>
> Anda harus menghapus catatan untuk domain yang Anda lepaskan dari server DNS Anda.

App Runner secara internal membuat sertifikat yang melacak validitas domain. Sertifikat ini disimpan di AWS Certificate Manager (ACM). App Runner tidak menghapus sertifikat ini selama 7 hari setelah domain dipisahkan dari layanan Anda atau setelah layanan dihapus.

Mengelola domain kustom

Mengelola domain kustom untuk layanan App Runner Anda menggunakan salah satu metode berikut:

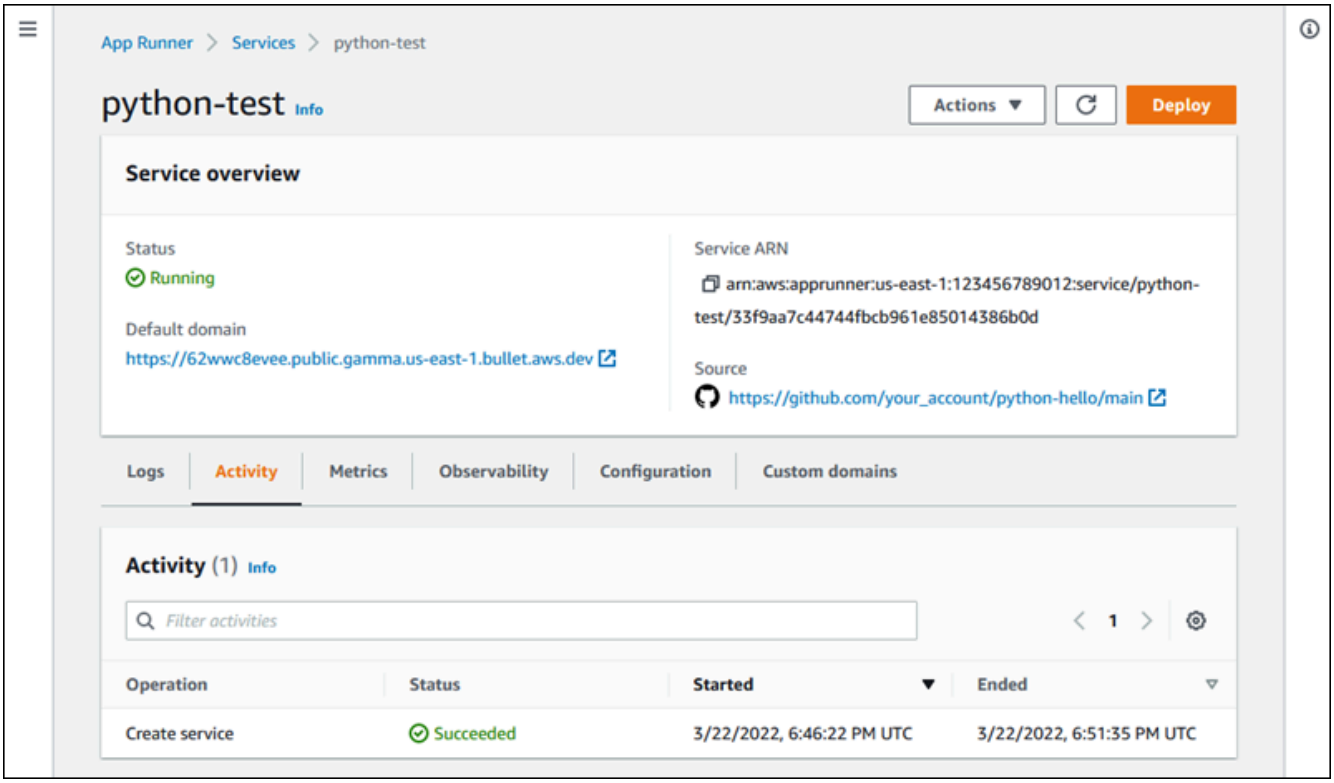
> **Note**
>
> Untuk kompatibilitas yang lebih baik dengan layanan App Runner, kami sarankan Anda menggunakan Amazon Route 53 sebagai penyedia DNS Anda. Jika Anda tidak menggunakan Amazon Route 53 untuk mengelola catatan DNS publik, hubungi penyedia DNS Anda untuk mengetahui cara menambahkan catatan.
> Jika Anda menggunakan Amazon Route 53 sebagai penyedia DNS, Anda dapat menambahkan CNAME atau alias record untuk subdomain. Untuk domain root, pastikan Anda menggunakan catatan alias.

App Runner console

Untuk mengaitkan (menautkan) domain kustom menggunakan konsol App Runner

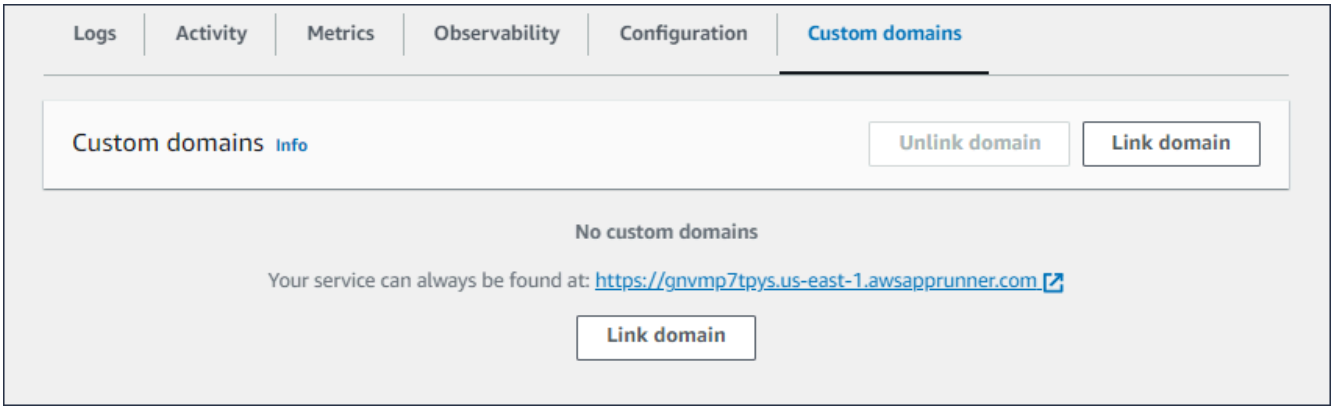
1. Buka konsol App Runner, dan di daftar Wilayah, pilih konsol Anda Wilayah AWS.
2. Di panel navigasi, pilih Layanan, lalu pilih layanan App Runner Anda.

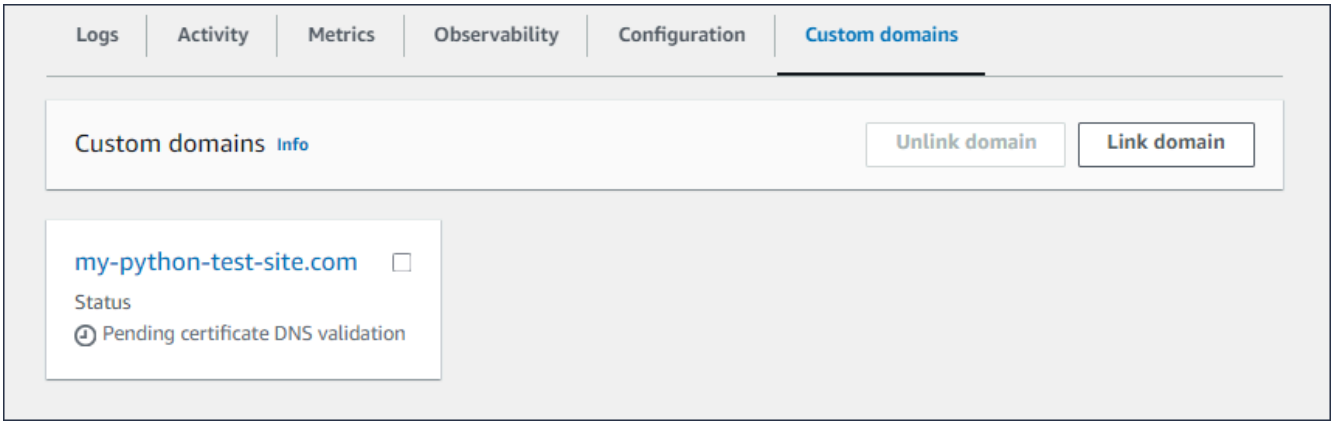
   Konsol menampilkan dasbor layanan dengan ikhtisar Layanan.



3. Pada halaman dasbor layanan, pilih tab Domain khusus.

   Konsol menampilkan domain kustom yang terkait dengan layanan Anda, atau Tidak ada domain khusus.

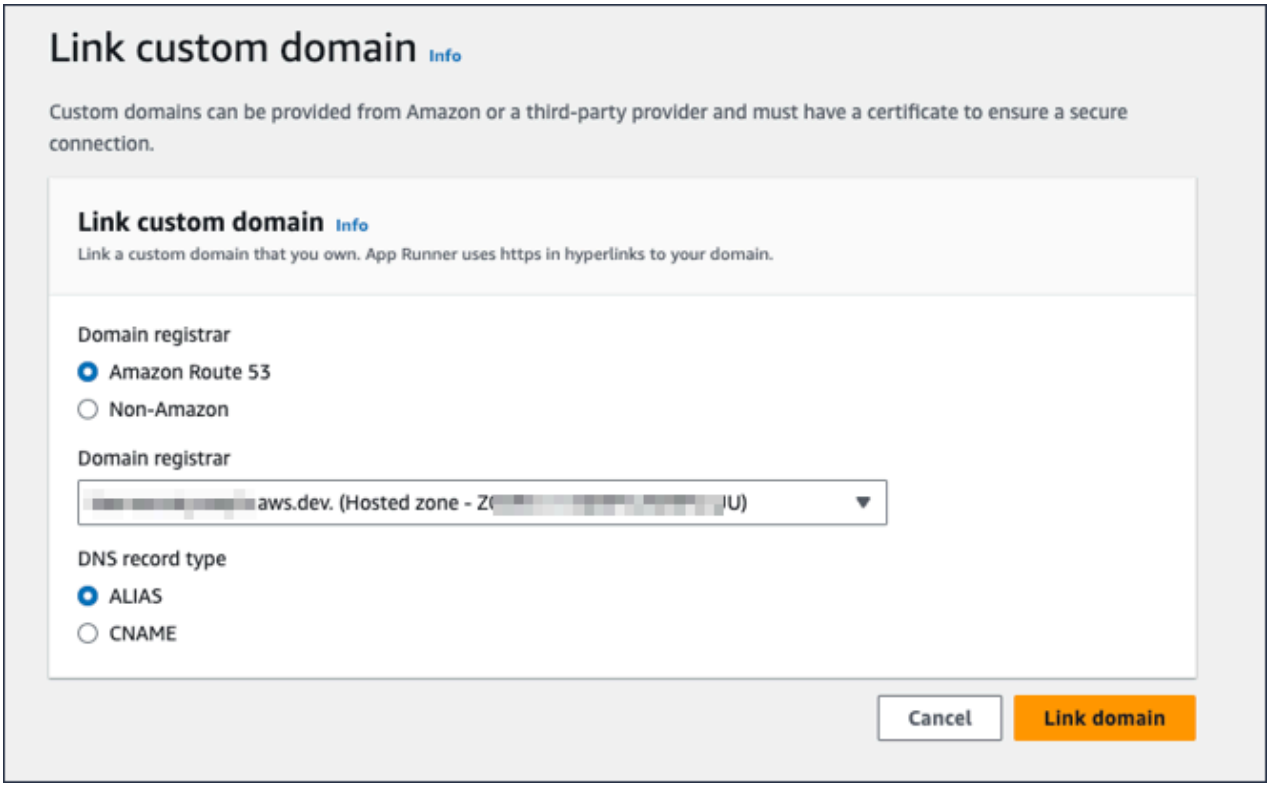




4. Pada tab Domain kustom, pilih Tautkan domain.
5. Halaman domain kustom Link ditampilkan.
   - Jika domain kustom Anda terdaftar di Amazon Route 53, pilih Amazon Route 53 untuk registrar Domain.
     a. Pilih nama Domain dari daftar drop-down. Daftar ini menampilkan nama nama domain Route 53 Anda dan id zona yang dihosting.

     > **Note**
     >
     > Anda harus terlebih dahulu membuat domain Route 53 menggunakan layanan Amazon Route 53 dari AWS akun yang sama yang Anda gunakan untuk mengelola sumber daya App Runner lainnya.

     b. Pilih jenis catatan DNS.
     c. Pilih Tautan domain.

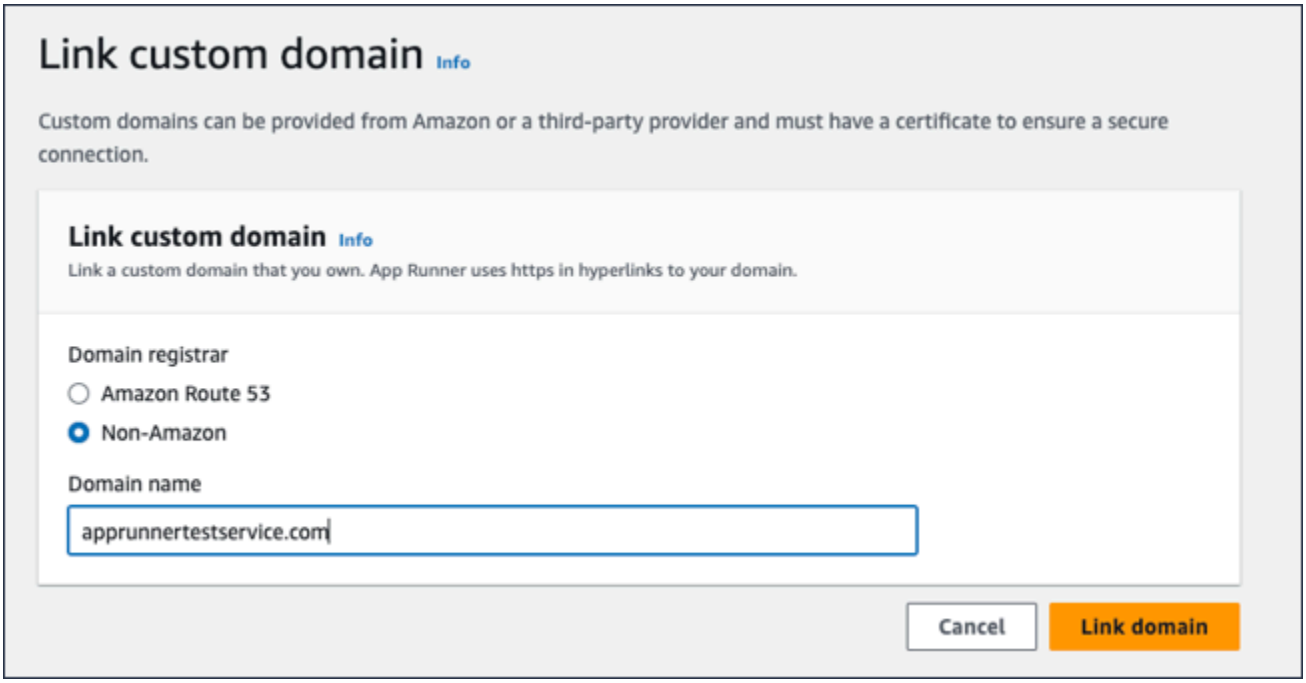


> **Note**
>
> Jika App Runner menampilkan pesan kesalahan yang menyatakan bahwa upaya konfigurasi otomatis gagal, Anda dapat melanjutkan dengan mengonfigurasi catatan DNS secara manual. Masalah ini dapat muncul jika nama domain yang sama sebelumnya tidak ditautkan dari layanan, tanpa catatan penyedia DNS yang mengarah ke layanan yang dihapus sesudahnya. Dalam hal ini App Runner diblokir agar tidak secara otomatis menimpa catatan ini. Untuk menyelesaikan konfigurasi DNS, lewati langkah-langkah lainnya dalam prosedur ini dan kemudian ikuti instruksi di. Konfigurasikan catatan alias Amazon Route 53

- Jika domain kustom Anda terdaftar dengan registrar domain lain, pilih Non-Amazon for Domain registrar.

  a. Masukkan nama Domain.

  b. Pilih Tautan domain.



6. Halaman Konfigurasi DNS ditampilkan.
   - Jika Amazon Route 53 penyedia DNS Anda, maka langkah ini opsional.

     Pada titik ini App Runner telah secara otomatis mengonfigurasi domain Route 53 Anda dengan validasi sertifikat dan catatan DNS yang diperlukan.

     > **Note**
     >
     > Jika nama domain yang sama ini sebelumnya tidak ditautkan dari layanan, tanpa catatan penyedia DNS yang mengarah ke layanan yang dihapus sesudahnya, konfigurasi otomatis yang dicoba oleh App Runner mungkin gagal. Untuk mengatasi masalah ini dan menyelesaikan asosiasi DNS, lanjutkan dengan langkah (1) dan (2) pada halaman Konfigurasi DNS untuk menyalin catatan target dan sertifikat saat ini ke penyedia DNS.

   - Salin catatan validasi sertifikat dan catatan target DNS, dan tambahkan ke server DNS Anda. App Runner kemudian dapat memvalidasi bahwa Anda memiliki atau mengontrol domain.

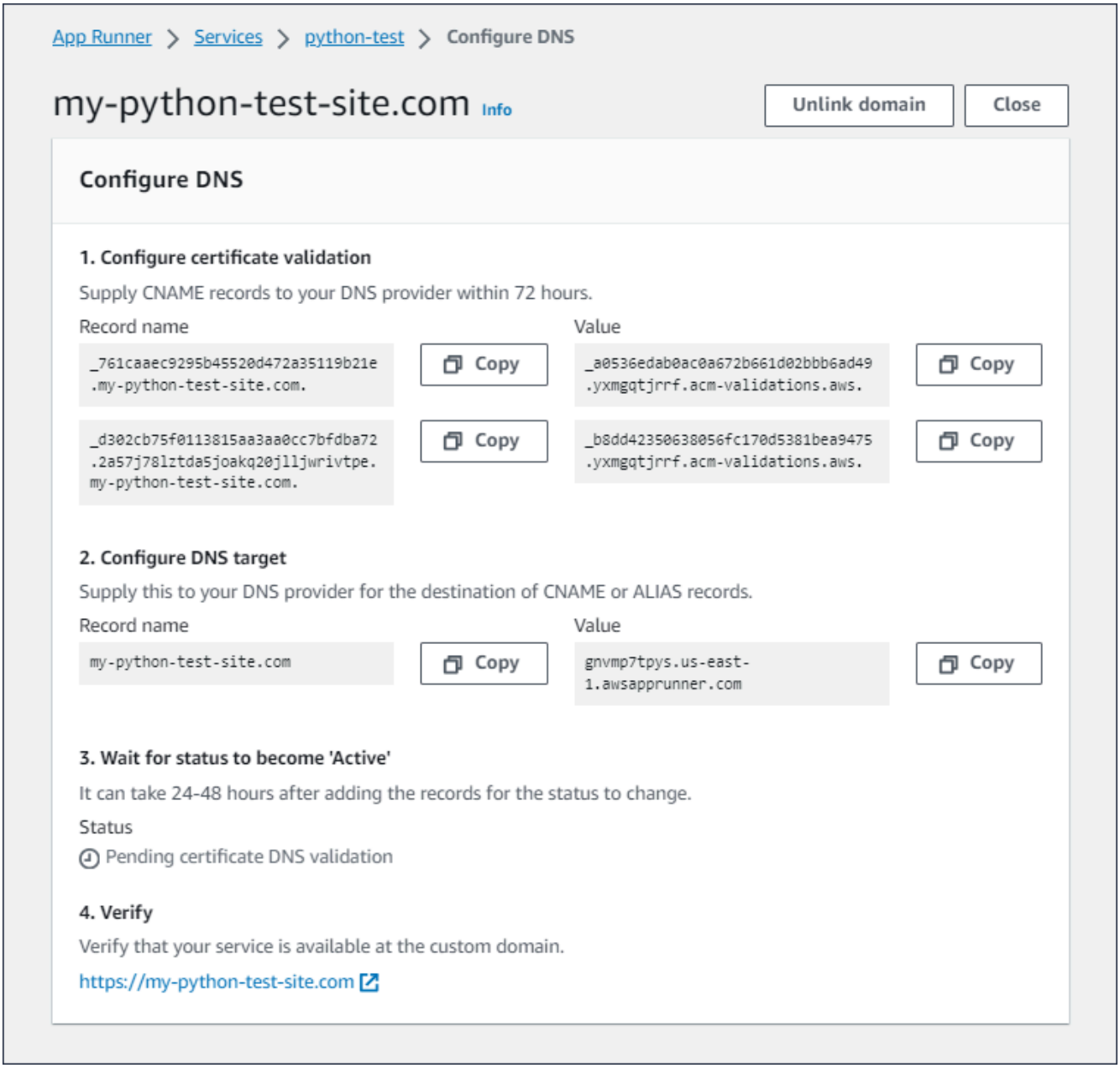
> **Note**
>
> Untuk memperbarui sertifikat domain kustom Anda secara otomatis, pastikan untuk tidak menghapus catatan validasi sertifikat dari server DNS Anda.

  - Untuk informasi selengkapnya tentang Mengonfigurasi validasi sertifikat, lihat Validasi DNS di Panduan Pengguna.AWS Certificate Manager
  - Untuk informasi tentang cara Mengonfigurasi target DNS dengan catatan alias Amazon Route 53, lihat. the section called “Konfigurasikan catatan alias Amazon Route 53”

- Jika Anda menggunakan penyedia DNS selain Amazon Route 53, ikuti langkah-langkah berikut.
  - Salin catatan validasi sertifikat dan catatan target DNS, dan tambahkan ke server DNS Anda. App Runner kemudian dapat memvalidasi bahwa Anda memiliki atau mengontrol domain.

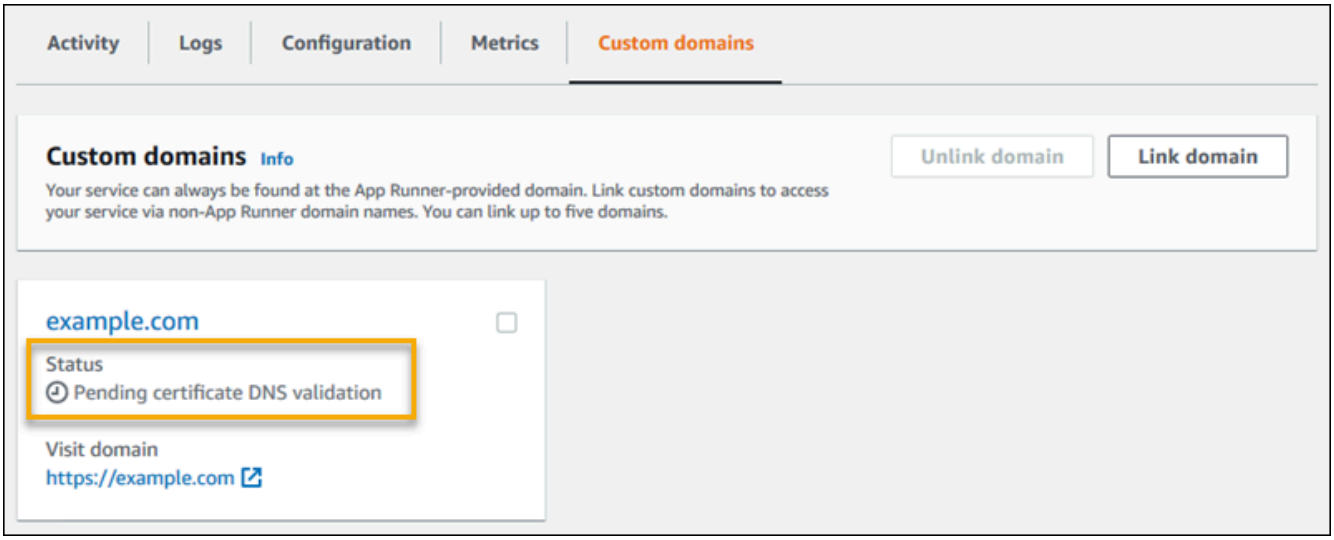
> **Note**
>
> Untuk memperbarui sertifikat domain kustom Anda secara otomatis, pastikan untuk tidak menghapus catatan validasi sertifikat dari server DNS Anda.

    - Untuk informasi selengkapnya tentang Mengonfigurasi validasi sertifikat, lihat Validasi DNS di Panduan Pengguna.AWS Certificate Manager
    - Untuk petunjuk tentang cara mengkonfigurasi target DNS pada pendaftar lain, seperti, Shopify GoDaddy, Hover, dan sebagainya, lihat dokumentasi spesifik mereka tentang menambahkan Target DNS.

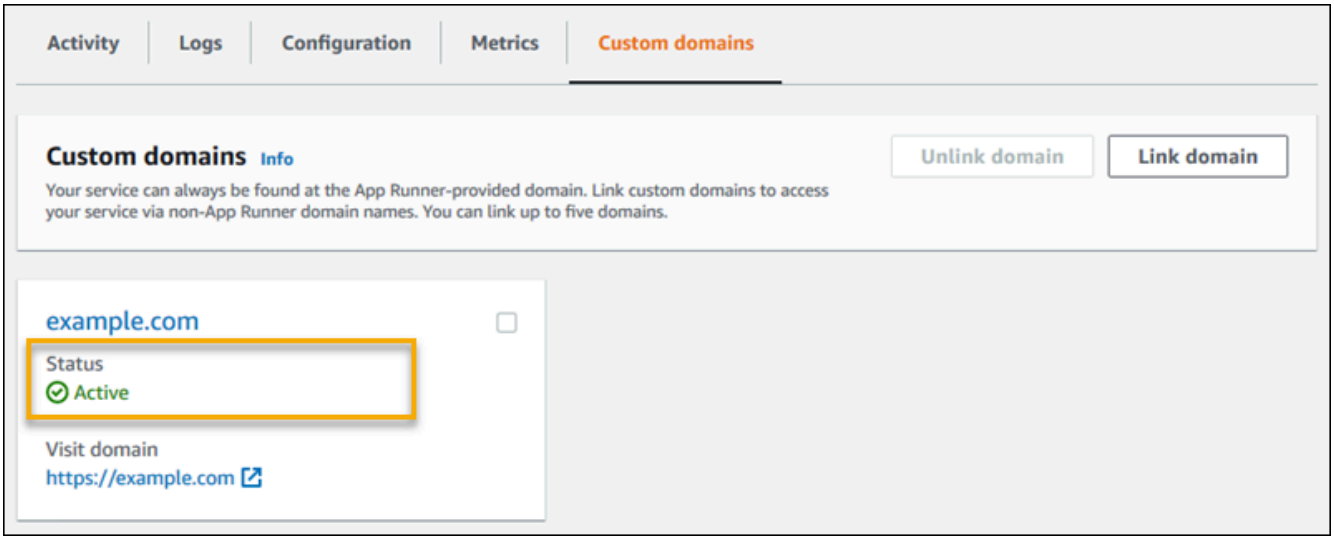


7. Pilih Tutup

Konsol menampilkan dasbor lagi. Tab Domain kustom memiliki ubin baru yang menunjukkan domain yang baru saja Anda tautkan dalam status validasi DNS sertifikat Tertunda.



8. Ketika status domain berubah menjadi Aktif, verifikasi bahwa domain berfungsi untuk merutekan lalu lintas dengan menjelajahinya.



> **Note**
>
> Untuk petunjuk tentang cara memecahkan masalah kesalahan yang terkait dengan domain kustom, lihat. the section called “Nama domain kustom”

Untuk memisahkan (memutuskan tautan) domain kustom menggunakan konsol App Runner

1. Pada tab Custom domains, pilih ubin untuk domain yang ingin Anda pisahkan, lalu pilih Unlink domain.

2. Dalam dialog Unlink domain, verifikasi tindakan dengan memilih Unlink domain.

> **Note**
>
> Anda harus menghapus catatan untuk domain yang Anda lepaskan dari server DNS Anda.

App Runner API or AWS CLI

Untuk mengaitkan domain kustom dengan layanan Anda menggunakan App Runner API atau AWS CLI, panggil tindakan AssociateCustomDomainAPI. Ketika panggilan berhasil, CustomDomainobjek dikembalikan yang menjelaskan domain kustom yang dikaitkan dengan layanan Anda. Objek menunjukkan `CREATING` status dan berisi daftar CertificateValidationRecordobjek. Panggilan juga mengembalikan alias target yang dapat Anda gunakan untuk mengkonfigurasi target DNS. Ini adalah catatan yang dapat Anda tambahkan ke DNS Anda.

Untuk memisahkan domain kustom dari layanan Anda menggunakan App Runner API atau AWS CLI, panggil tindakan DisassociateCustomDomainAPI. Ketika panggilan berhasil, CustomDomainobjek dikembalikan yang menjelaskan domain kustom yang sedang dipisahkan dari layanan Anda. Objek menunjukkan `DELETING` status.

Topik

- Konfigurasikan catatan alias Amazon Route 53 untuk DNS target Anda

Konfigurasikan catatan alias Amazon Route 53 untuk DNS target Anda

> **Note**
>
> Anda tidak perlu mengikuti prosedur ini jika Amazon Route 53 adalah penyedia DNS Anda. Dalam hal ini App Runner secara otomatis mengonfigurasi domain Route 53 Anda dengan validasi sertifikat yang diperlukan dan catatan DNS untuk menautkan ke aplikasi web App Runner Anda.
> Jika upaya konfigurasi otomatis App Runner gagal, ikuti prosedur ini untuk menyelesaikan konfigurasi DNS. Jika nama domain yang sama sebelumnya tidak ditautkan dari layanan, tanpa catatan penyedia DNS yang mengarah ke layanan yang dihapus sesudahnya, App

Runner diblokir agar tidak secara otomatis menimpa catatan ini. Prosedur ini menjelaskan cara menyalinnya secara manual ke DNS Route 53 Anda.

Anda dapat menggunakan Amazon Route 53 sebagai penyedia DNS untuk merutekan lalu lintas ke layanan App Runner Anda. Ini adalah layanan web Domain Name System (DNS) yang sangat tersedia dan terukur. Catatan Amazon Route 53 berisi pengaturan yang mengontrol cara lalu lintas dialihkan ke layanan App Runner Anda. Anda membuat catatan CNAME atau catatan ALIAS. Untuk perbandingan pada catatan CNAME dan alias, lihat Memilih antara catatan alias dan non-alias, di Panduan Pengembang Amazon Route 53.

> **Note**
>
> Amazon Route 53 saat ini mendukung catatan alias untuk layanan yang dibuat setelah 1 Agustus 2022.

Amazon Route 53 console

Untuk mengonfigurasi catatan alias Amazon Route 53

1. Masuk ke AWS Management Console dan buka konsol Route 53.
2. Pada panel navigasi, pilih Zona yang di-hosting.
3. Pilih nama zona yang dihosting yang ingin Anda gunakan untuk merutekan lalu lintas ke layanan App Runner Anda.
4. Pilih Create record (Buat catatan).
5. Tentukan nilai-nilai berikut ini:
   - Kebijakan perutean: Pilih kebijakan perutean yang berlaku. Untuk informasi selengkapnya, lihat Memilih kebijakan perutean.
   - Nama rekaman: Masukkan nama domain yang ingin Anda gunakan untuk merutekan lalu lintas ke layanan App Runner Anda. Nilai default adalah nama zona yang di-hosting. Misalnya, jika nama zona yang dihosting adalah `example.com` dan Anda ingin menggunakan `acme.example.com` untuk merutekan lalu lintas ke lingkungan Anda, masukkan`acme`.
   - Nilai/Rute lalu lintas ke: Pilih Alias ke Aplikasi Pelari Aplikasi, lalu pilih Wilayah tempat titik akhir berasal. Pilih nama domain aplikasi yang ingin Anda rutekan lalu lintas.

- Jenis rekaman: Terima IPv4 alamat default, A —.
- Evaluasi kesehatan target: Terima nilai default, Ya.

6. Pilih Create records (Buat catatan).

Catatan alias Route 53 yang Anda buat akan disebarkan di semua server Route 53 dalam waktu 60 detik. Ketika server Route 53 disebarkan dengan catatan alias Anda, Anda dapat merutekan lalu lintas ke layanan App Runner dengan menggunakan nama catatan alias yang Anda buat.

Untuk informasi tentang cara memecahkan masalah jika perubahan DNS terlalu lama untuk disebarkan, lihat Mengapa perubahan DNS saya membutuhkan waktu lama untuk menyebar di Route 53 dan resolver publik? .

Amazon Route 53 API or AWS CLI

Untuk mengonfigurasi rekaman alias Amazon Route 53 menggunakan Amazon Route 53 API atau AWS CLI memanggil tindakan ChangeResourceRecordSetsAPI. Untuk mempelajari tentang id zona yang dihosting target dari Route 53, lihat Titik akhir layanan.

Menjeda dan melanjutkan layanan App Runner

Jika Anda perlu menonaktifkan aplikasi web Anda sementara dan menghentikan kode agar tidak berjalan, Anda dapat menjeda AWS App Runner layanan Anda. App Runner mengurangi kapasitas komputasi untuk layanan menjadi nol.

Ketika Anda siap untuk menjalankan aplikasi Anda lagi, Anda dapat melanjutkan layanan App Runner Anda. App Runner menyediakan kapasitas komputasi baru, menyebarkan aplikasi Anda ke dalamnya, dan menjalankan aplikasi. Sumber aplikasi Anda tidak di-deploy ulang, dan tidak diperlukan build. Sebaliknya, App Runner melanjutkan dengan versi yang Anda gunakan saat ini. Aplikasi Anda mempertahankan domain App Runner-nya.

> **Important**
>
> - Ketika Anda menjeda layanan Anda, aplikasi Anda kehilangan statusnya. Misalnya, penyimpanan sementara apa pun yang kode Anda gunakan hilang. Untuk kode Anda, menjeda dan melanjutkan layanan Anda sama dengan menerapkan ke layanan baru.

- Jika Anda menjeda layanan karena cacat pada kode Anda (misalnya, bug atau masalah keamanan yang ditemukan), Anda tidak dapat menerapkan versi baru sebelum melanjutkan layanan.

  Oleh karena itu, kami menyarankan agar Anda tetap menjalankan layanan dan memutar kembali ke versi aplikasi stabil terakhir Anda.
- Saat Anda melanjutkan layanan, App Runner akan menerapkan versi aplikasi terakhir yang digunakan sebelum Anda menjeda layanan. Jika Anda menambahkan versi sumber baru sejak menjeda layanan, App Runner tidak akan menerapkannya secara otomatis meskipun penerapan otomatis dipilih. Misalnya, anggap Anda memiliki versi gambar baru di repositori gambar atau komit baru di repositori kode. Versi ini tidak digunakan secara otomatis.

  Untuk menerapkan versi yang lebih baru, lakukan penerapan manual atau tambahkan versi lain ke repositori sumber Anda setelah melanjutkan layanan App Runner.

Menjeda dan menghapus dibandingkan

Jeda layanan App Runner Anda untuk menonaktifkannya sementara. Hanya sumber daya komputasi yang dihentikan, dan data yang Anda simpan (misalnya, gambar kontainer dengan versi aplikasi Anda) tetap utuh. Melanjutkan layanan Anda dengan cepat—aplikasi Anda siap digunakan ke sumber daya komputasi baru. Domain App Runner Anda tetap sama.

Hapus layanan App Runner Anda untuk menghapusnya secara permanen. Data yang Anda simpan akan dihapus. Jika Anda perlu membuat ulang layanan, App Runner perlu mengambil sumber Anda lagi, dan juga membangunnya jika itu adalah repositori kode. Aplikasi web Anda mendapatkan domain App Runner baru.

Saat layanan Anda dijeda

Saat Anda menjeda layanan dan berada dalam status Dijeda, layanan akan merespons permintaan tindakan secara berbeda, termasuk panggilan API atau operasi konsol. Saat layanan dijeda, Anda masih dapat melakukan tindakan App Runner yang tidak mengubah definisi atau konfigurasi layanan dengan cara yang memengaruhi runtime. Dengan kata lain, jika suatu tindakan mengubah perilaku, skala, atau karakteristik lain dari layanan yang sedang berjalan, Anda tidak dapat melakukan tindakan tersebut pada layanan yang dijeda.

Daftar berikut memberikan informasi tentang tindakan API yang dapat dan tidak dapat Anda lakukan pada layanan yang dijeda. Operasi konsol yang setara juga diizinkan atau ditolak.

Tindakan yang dapat Anda lakukan pada layanan yang dijeda

- `List*`dan `Describe*` tindakan — Tindakan yang hanya membaca informasi.
- `DeleteService`— Anda selalu dapat menghapus layanan.
- `TagResource`Tag dikaitkan dengan layanan, tetapi bukan bagian dari definisinya dan tidak memengaruhi perilaku runtime-nya. `UntagResource`

Tindakan yang tidak dapat Anda lakukan pada layanan yang dijeda

- `StartDeployment`tindakan (atau penerapan manual menggunakan konsol)
- `UpdateService`(atau perubahan konfigurasi menggunakan konsol, kecuali untuk perubahan penandaan)
- `CreateCustomDomainAssociations`, `DeleteCustomDomainAssociations`
- `CreateConnection`, `DeleteConnection`

Jeda dan lanjutkan layanan Anda

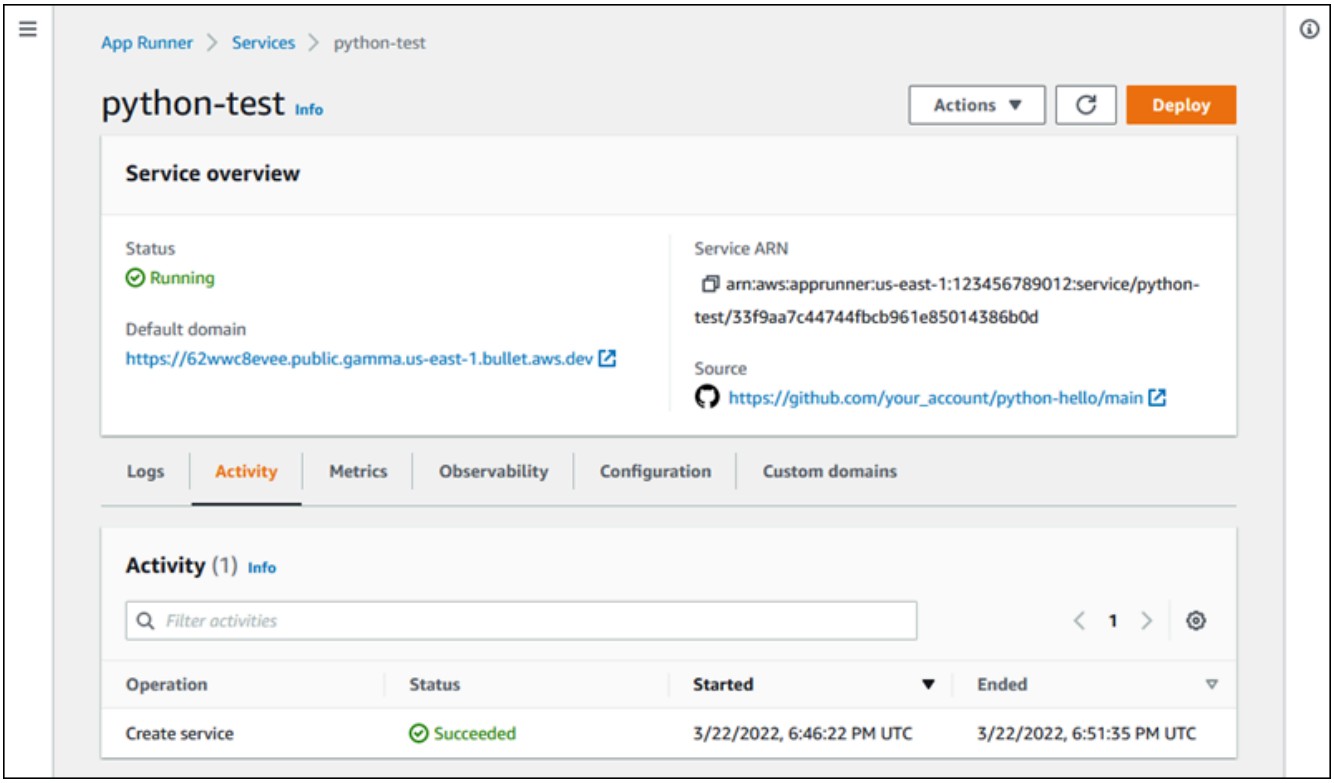
Jeda dan lanjutkan layanan App Runner menggunakan salah satu metode berikut:

App Runner console

Untuk menjeda layanan Anda menggunakan konsol App Runner

1. Buka konsol App Runner, dan di daftar Wilayah, pilih konsol Anda Wilayah AWS.
2. Di panel navigasi, pilih Layanan, lalu pilih layanan App Runner Anda.

   Konsol menampilkan dasbor layanan dengan ikhtisar Layanan.



3. Pilih Tindakan, lalu pilih Jeda.

   Pada halaman dasbor layanan, Status layanan berubah menjadi Operasi yang sedang berlangsung, dan kemudian berubah menjadi Dijeda. Layanan Anda sekarang dijeda.

Untuk melanjutkan layanan Anda menggunakan konsol App Runner

1. Pilih Tindakan, lalu pilih Lanjutkan.

   Pada halaman dasbor layanan, Status layanan berubah menjadi Operasi yang sedang berlangsung.

2. Tunggu layanan dilanjutkan. Pada halaman dasbor layanan, Status layanan berubah kembali ke Running.

3. Untuk memverifikasi bahwa melanjutkan layanan berhasil, pada halaman dasbor layanan, pilih nilai domain App Runner. Ini adalah URL untuk situs web layanan Anda. Verifikasi bahwa aplikasi web Anda berjalan dengan benar.

App Runner API or AWS CLI

Untuk menjeda layanan Anda menggunakan App Runner API atau AWS CLI, panggil tindakan PauseServiceAPI. Jika panggilan mengembalikan respons yang berhasil dengan objek Service yang ditampilkan`"Status": "OPERATION_IN_PROGRESS"`, App Runner mulai menjeda layanan Anda.

Untuk melanjutkan layanan Anda menggunakan App Runner API atau AWS CLI, panggil tindakan ResumeServiceAPI. Jika panggilan mengembalikan respons yang berhasil dengan objek Service yang ditampilkan`"Status": "OPERATION_IN_PROGRESS"`, App Runner mulai melanjutkan layanan Anda.

Menghapus layanan App Runner

Ketika Anda ingin menghentikan aplikasi web yang berjalan di AWS App Runner layanan Anda, Anda dapat menghapus layanan. Menghapus layanan menghentikan layanan web yang sedang berjalan, menghapus sumber daya yang mendasarinya, dan menghapus data terkait Anda.

Anda mungkin ingin menghapus layanan App Runner karena satu atau beberapa alasan berikut:

- Anda tidak memerlukan aplikasi web lagi — Misalnya, sudah pensiun, atau versi pengembangan yang sudah selesai Anda gunakan.
- Anda telah mencapai kuota layanan App Runner — Anda ingin membuat layanan baru yang sama Wilayah AWS dan Anda telah mencapai kuota yang terkait dengan akun Anda. Untuk informasi selengkapnya, lihat the section called “Kuota sumber daya App Runner”.
- Pertimbangan keamanan atau privasi — Anda ingin App Runner menghapus data yang disimpan untuk layanan Anda.

Menjeda dan menghapus dibandingkan

Jeda layanan App Runner Anda untuk menonaktifkannya sementara. Hanya sumber daya komputasi yang dihentikan, dan data yang Anda simpan (misalnya, gambar kontainer dengan versi aplikasi Anda) tetap utuh. Melanjutkan layanan Anda dengan cepat—aplikasi Anda siap digunakan ke sumber daya komputasi baru. Domain App Runner Anda tetap sama.

Hapus layanan App Runner Anda untuk menghapusnya secara permanen. Data yang Anda simpan akan dihapus. Jika Anda perlu membuat ulang layanan, App Runner perlu mengambil sumber Anda

lagi, dan juga membangunnya jika itu adalah repositori kode. Aplikasi web Anda mendapatkan domain App Runner baru.

Apa yang dihapus oleh App Runner?

Saat Anda menghapus layanan, App Runner menghapus beberapa item terkait, dan tidak menghapus yang lain. Daftar berikut memberikan detailnya.

Item yang dihapus oleh App Runner:

- Gambar kontainer — Salinan gambar yang Anda gunakan atau gambar yang dibuat oleh App Runner dari kode sumber Anda. Ini disimpan di Amazon Elastic Container Registry (Amazon ECR) Registry (Amazon ECR) menggunakan Akun AWS internal yang dimiliki oleh App Runner.
- Konfigurasi layanan — Pengaturan konfigurasi yang terkait dengan layanan App Runner Anda. Mereka disimpan di Amazon DynamoDB menggunakan Akun AWS internal yang dimiliki oleh App Runner.

Item yang tidak dihapus oleh App Runner:

- Koneksi — Anda mungkin memiliki koneksi yang terkait dengan layanan Anda. Koneksi App Runner adalah sumber daya terpisah yang mungkin dibagikan di antara beberapa layanan App Runner. Jika Anda tidak memerlukan koneksi lagi, Anda dapat menghapusnya secara eksplisit. Untuk informasi selengkapnya, lihat the section called “Koneksi”.
- Sertifikat domain khusus — Jika Anda menautkan domain kustom ke layanan App Runner, App Runner secara internal membuat sertifikat yang melacak validitas domain. Mereka disimpan di AWS Certificate Manager (ACM). App Runner tidak menghapus sertifikat selama tujuh hari setelah domain dibatalkan tautannya dari layanan Anda atau setelah layanan dihapus. Untuk informasi selengkapnya, lihat the section called “Nama domain khusus”.

Hapus layanan Anda

Hapus layanan App Runner Anda menggunakan salah satu metode berikut:

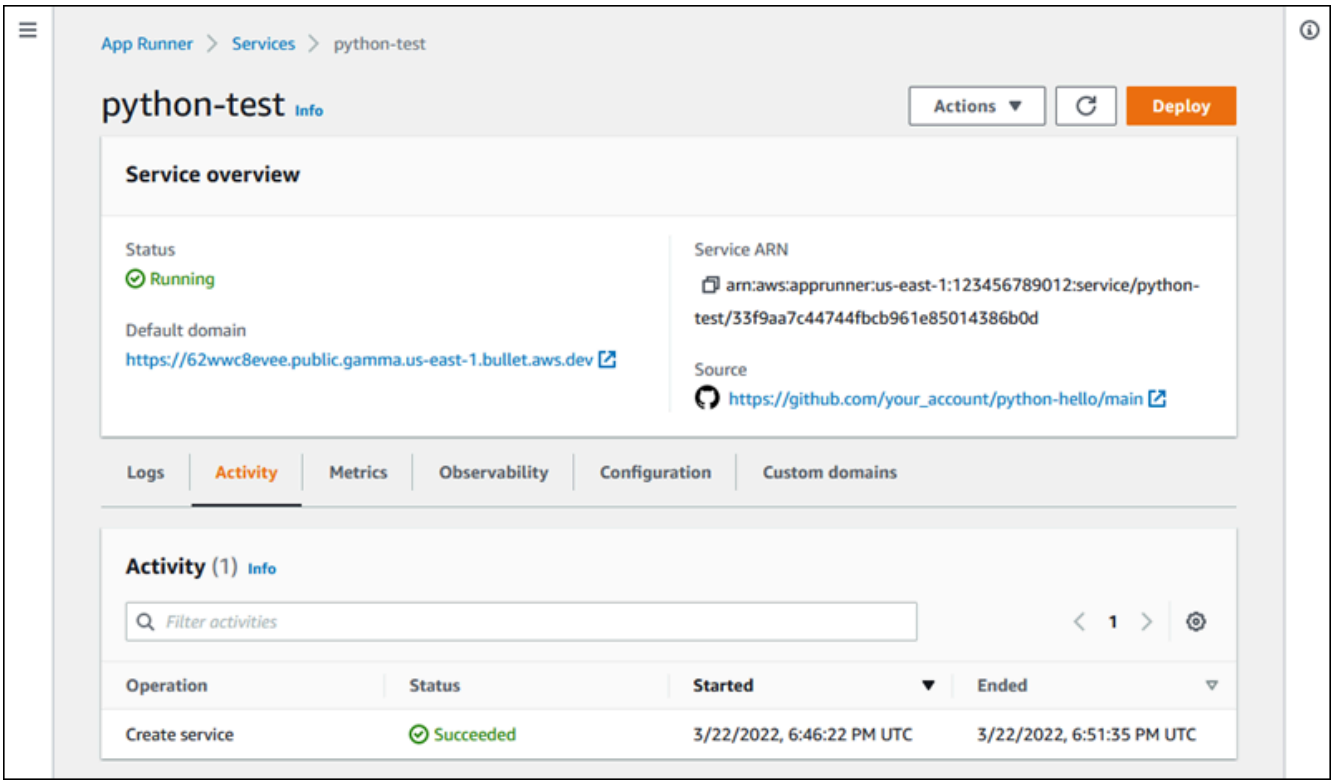
App Runner console

Untuk menghapus layanan Anda menggunakan konsol App Runner

1. Buka konsol App Runner, dan di daftar Wilayah, pilih konsol Anda Wilayah AWS.

2. Di panel navigasi, pilih Layanan, lalu pilih layanan App Runner Anda.

   Konsol menampilkan dasbor layanan dengan ikhtisar Layanan.



3. Pilih Tindakan, lalu pilih Hapus.

   Konsol membawa Anda ke halaman Layanan. Layanan yang dihapus menampilkan status Operasi dalam proses, dan kemudian layanan menghilang dari daftar. Layanan Anda sekarang dihapus.

App Runner API or AWS CLI

Untuk menghapus layanan Anda menggunakan App Runner API atau AWS CLI, panggil tindakan DeleteServiceAPI. Jika panggilan mengembalikan respons yang berhasil dengan objek Service yang ditampilkan`"Status": "OPERATION_IN_PROGRESS"`, App Runner mulai menghapus layanan Anda.

Merujuk variabel lingkungan

Dengan App Runner, Anda dapat mereferensikan rahasia dan konfigurasi sebagai variabel lingkungan dalam layanan Anda saat membuat layanan atau memperbarui layanan.

Anda dapat mereferensikan data konfigurasi yang tidak sensitif seperti batas waktu dan hitungan coba lagi dalam Teks Biasa sebagai pasangan nilai kunci. Data konfigurasi yang Anda referensikan dalam Teks Biasa tidak dienkripsi dan dapat dilihat oleh orang lain dalam konfigurasi layanan App Runner dan log aplikasi.

> **Note**
>
> Untuk alasan keamanan, jangan mereferensikan data sensitif apa pun dalam Teks Biasa di layanan App Runner Anda.

Mereferensikan data sensitif sebagai variabel lingkungan

App Runner mendukung referensi data sensitif secara aman sebagai variabel lingkungan di layanan Anda. Pertimbangkan untuk menyimpan data sensitif yang ingin Anda referensikan di AWS Secrets Manageratau AWS Systems Manager Parameter Store. Kemudian, Anda dapat mereferensikannya dengan aman di layanan Anda sebagai variabel lingkungan dari konsol App Runner atau dengan memanggil API. Ini secara efektif memisahkan manajemen rahasia dan parameter dari kode aplikasi dan konfigurasi layanan Anda, meningkatkan keamanan keseluruhan aplikasi Anda yang berjalan di App Runner.

> **Note**
>
> App Runner tidak mengenakan biaya untuk mereferensikan Secrets Manager dan SSM Parameter Store sebagai variabel lingkungan. Namun, Anda membayar harga standar untuk menggunakan Secrets Manager dan SSM Parameter Store.
> Untuk informasi selengkapnya tentang harga, lihat berikut ini:
>
> - AWS Harga Secrets Manager
> - AWS Harga Toko Parameter SSM

Berikut ini adalah proses untuk mereferensikan data sensitif sebagai variabel lingkungan:

1. Menyimpan data sensitif, seperti kunci API, kredensyal database, parameter koneksi database, atau versi aplikasi sebagai rahasia atau parameter di salah satu AWS Secrets Manager atau AWS Systems Manager Parameter Store.
2. Perbarui kebijakan IAM peran instans Anda agar App Runner dapat mengakses rahasia dan parameter yang disimpan di Secrets Manager dan SSM Parameter Store. Untuk informasi selengkapnya, lihat Izin.
3. Referensikan rahasia dan parameter secara aman sebagai variabel lingkungan dengan menetapkan nama dan memberikan Nama Sumber Daya Amazon (ARN) mereka. Anda dapat menambahkan variabel lingkungan saat membuat layanan atau memperbarui konfigurasi layanan. Anda dapat menggunakan salah satu opsi berikut untuk menambahkan variabel lingkungan:
   - Konsol Pelari Aplikasi
   - API Pelari Aplikasi
   - `apprunner.yaml`berkas konfigurasi

   > **Note**
   >
   > Anda tidak dapat menetapkan `PORT` sebagai nama untuk variabel lingkungan saat membuat atau memperbarui layanan App Runner Anda. Ini adalah variabel lingkungan yang dicadangkan untuk layanan App Runner.

   Untuk informasi selengkapnya tentang cara mereferensikan rahasia dan parameter, lihat Mengelola variabel lingkungan.

> **Note**
>
> Karena App Runner hanya menyimpan referensi ke rahasia dan parameter ARNs, data sensitif tidak terlihat oleh orang lain dalam konfigurasi layanan App Runner dan log aplikasi.

Pertimbangan

- Pastikan Anda memperbarui peran instans dengan izin yang sesuai untuk mengakses rahasia dan parameter di AWS Secrets Manager atau di AWS Systems Manager Parameter Store. Untuk informasi selengkapnya, lihat Izin.
- Pastikan AWS Systems Manager Parameter Store Akun AWS sama dengan layanan yang ingin Anda luncurkan atau perbarui. Saat ini, Anda tidak dapat mereferensikan parameter Penyimpanan Parameter SSM di seluruh akun.
- Ketika rahasia dan nilai parameter diputar atau diubah, nilai tersebut tidak diperbarui secara otomatis di layanan App Runner Anda. Menerapkan ulang layanan App Runner Anda karena App Runner hanya menarik rahasia dan parameter selama penerapan.
- Anda juga memiliki opsi untuk langsung memanggil AWS Secrets Manager dan AWS Systems Manager Parameter Store melalui SDK di layanan App Runner Anda.
- Untuk menghindari kesalahan, pastikan hal berikut saat mereferensikannya sebagai variabel lingkungan:
  - Anda menentukan ARN rahasia yang tepat.
  - Anda menentukan nama yang tepat atau ARN dari parameter.

Izin

Untuk mengaktifkan referensi rahasia dan parameter yang disimpan di Penyimpanan Parameter SSM AWS Secrets Manager atau SSM, tambahkan izin yang sesuai ke kebijakan IAM peran instans Anda untuk mengakses Secrets Manager dan SSM Parameter Store.

> **Note**
>
> App Runner tidak dapat mengakses sumber daya di akun Anda tanpa izin Anda. Anda memberikan izin melalui memperbarui kebijakan IAM Anda.

Anda dapat menggunakan templat kebijakan berikut untuk memperbarui peran instans di konsol IAM. Anda dapat memodifikasi templat kebijakan ini untuk memenuhi kebutuhan spesifik Anda. Untuk informasi selengkapnya tentang memperbarui peran instance, lihat Memodifikasi peran dalam Panduan Pengguna IAM.

> **Note**
>
> Anda juga dapat menyalin template berikut dari konsol App Runner saat membuat variabel lingkungan.

Salin, templat berikut ke peran instans Anda untuk menambahkan izin ke rahasia referensi dari AWS Secrets Manager.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue",
        "kms:Decrypt*"
      ],
      "Resource": [
        "arn:aws:secretsmanager:<region>:<aws_account_id>:secret:<secret_name>",
        "arn:aws:kms:<region>:<aws_account_id>:key/<key_id>"
      ]
    }
  ]
}
```

Salin template berikut ke peran instans Anda untuk menambahkan izin ke parameter referensi dari AWS Systems ManagerParameter Store.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:GetParameters"
      ],
      "Resource": [
        "arn:aws:ssm:<region>:<aws_account_id>:parameter/<parameter_name>"
      ]
    }
```

```
  ]
}
```

Mengelola variabel lingkungan Anda

Mengelola variabel lingkungan untuk layanan App Runner Anda dengan menggunakan salah satu metode berikut:

- the section called “Konsol Pelari Aplikasi”
- the section called “API Pelari Aplikasi atau AWS CLI”

Konsol Pelari Aplikasi

Saat membuat layanan atau memperbarui layanan di konsol App Runner, Anda dapat menambahkan variabel lingkungan.

Menambahkan variabel lingkungan

Untuk menambahkan variabel lingkungan

1. Buka konsol App Runner, dan di daftar Wilayah, pilih konsol Anda Wilayah AWS.
2. Berdasarkan apakah Anda membuat atau memperbarui layanan, lakukan salah satu langkah berikut:
   - Jika Anda membuat layanan baru, pilih Buat layanan Pelari Aplikasi dan buka Konfigurasi Layanan.
   - Jika Anda memperbarui layanan yang ada, pilih layanan yang ingin Anda perbarui dan buka tab Konfigurasi layanan.
3. Pergi ke variabel Lingkungan - opsional di bawah Pengaturan layanan.
4. Pilih salah satu opsi berikut berdasarkan kebutuhan Anda:
   - Pilih Teks Biasa dari sumber variabel Lingkungan dan masukkan pasangan nilai kunci-nya di bawah nama variabel Lingkungan dan nilai variabel Lingkungan, masing-masing.

> **Note**
>
> Pilih Teks Biasa jika Anda ingin mereferensikan data yang tidak sensitif. Data ini tidak dienkripsi dan dapat dilihat oleh orang lain dalam konfigurasi layanan App Runner dan log aplikasi.

- Pilih Secrets Manager dari sumber variabel Lingkungan untuk mereferensikan rahasia yang disimpan AWS Secrets Manager sebagai variabel lingkungan dalam layanan Anda. Berikan nama variabel lingkungan dan Amazon Resource Name (ARN) dari rahasia yang Anda referensikan di bawah Environment variable name dan Environment variable value masing-masing.
- Pilih Penyimpanan Parameter SSM dari sumber variabel Lingkungan untuk mereferensikan parameter yang disimpan di Penyimpanan Parameter SSM sebagai variabel lingkungan dalam layanan Anda. Berikan nama variabel lingkungan dan ARN dari parameter yang Anda referensikan di bawah nama variabel Lingkungan dan nilai variabel Lingkungan masing-masing.

> **Note**
>
> - Anda tidak dapat menetapkan `PORT` sebagai nama untuk variabel lingkungan saat membuat atau memperbarui layanan App Runner Anda. Ini adalah variabel lingkungan yang dicadangkan untuk layanan App Runner.
> - Jika parameter Penyimpanan Parameter SSM Wilayah AWS sama dengan layanan yang ingin Anda luncurkan, Anda dapat menentukan Nama Sumber Daya Amazon (ARN) lengkap atau nama parameter. Jika parameter berada di wilayah yang berbeda, Anda perlu menentukan ARN lengkap.
> - Pastikan parameter yang Anda referensikan berada di akun yang sama dengan layanan yang Anda luncurkan atau perbarui. Saat ini, Anda tidak dapat mereferensikan parameter Penyimpanan Parameter SSM di seluruh akun.

5. Pilih Tambahkan variabel lingkungan untuk referensi ke variabel lingkungan lain.
6. Perluas templat kebijakan IAM untuk melihat dan menyalin templat kebijakan IAM yang disediakan untuk AWS Secrets Manager dan Penyimpanan Parameter SSM. Anda hanya perlu melakukan ini jika Anda belum memperbarui kebijakan IAM dari peran instans Anda dengan izin yang diperlukan. Untuk informasi selengkapnya, lihat Izin.

Menghapus variabel lingkungan

Sebelum Anda menghapus variabel lingkungan, pastikan kode aplikasi Anda diperbarui untuk mencerminkan hal yang sama. Jika kode aplikasi tidak diperbarui, layanan App Runner Anda mungkin gagal.

Untuk menghapus variabel lingkungan

1. Buka konsol App Runner, dan di daftar Wilayah, pilih konsol Anda Wilayah AWS.
2. Buka tab Konfigurasi layanan yang ingin Anda perbarui.
3. Pergi ke variabel Lingkungan - opsional di bawah Pengaturan layanan.
4. Pilih Hapus di samping variabel lingkungan yang ingin Anda hapus. Anda menerima pesan untuk mengonfirmasi penghapusan.
5. Pilih Hapus.

API Pelari Aplikasi atau AWS CLI

Anda dapat mereferensikan data sensitif yang disimpan di Secrets Manager dan SSM Parameter Store dengan menambahkannya sebagai variabel lingkungan dalam layanan Anda.

> **Note**
>
> Perbarui kebijakan IAM peran instans Anda agar App Runner dapat mengakses rahasia dan parameter yang disimpan di Secrets Manager dan SSM Parameter Store. Untuk informasi selengkapnya, lihat Izin.

Untuk mereferensikan rahasia dan konfigurasi sebagai variabel lingkungan

1. Buat rahasia atau konfigurasi di Secrets Manager atau SSM Parameter Store.

   Contoh berikut menunjukkan cara membuat rahasia dan parameter menggunakan SSM Parameter Store.

   Example Membuat rahasia - Permintaan

   Contoh berikut menunjukkan cara membuat rahasia yang mewakili kredensi database.

   ```
   aws secretsmanager create-secret \
   ```

```
–name DevRdsCredentials \
–description "Rds credentials for development account." \
–secret-string "{\"user\":\"diegor\",\"password\":\"EXAMPLE-PASSWORD\"}"
```

Example Membuat rahasia - Respon

```
arn:aws:secretsmanager:<region>:<aws_account_id>:secret:DevRdsCredentials
```

Example Membuat konfigurasi - Permintaan

Contoh berikut menunjukkan cara membuat parameter yang mewakili string koneksi RDS.

```
aws systemsmanager put-parameter \
–name DevRdsConnectionString \
–value "mysql2://dev-mysqlcluster-rds.com:3306/diegor" \
–type "String" \
–description "Rds connection string for development account."
```

Example Membuat konfigurasi - Respon

```
arn:aws:ssm:<region>:<aws_account_id>:parameter/DevRdsConnectionString
```

2. Referensikan rahasia dan konfigurasi yang disimpan di Secrets Manager dan SSM Parameter Store dengan menambahkannya sebagai variabel lingkungan. Anda dapat menambahkan variabel lingkungan saat membuat atau memperbarui layanan App Runner.

   Contoh berikut menunjukkan cara mereferensikan rahasia dan konfigurasi sebagai variabel lingkungan pada layanan App Runner berbasis kode dan berbasis gambar.

   Example Masukan file.json untuk layanan App Runner berbasis gambar

```
{
  "ServiceName": "example-secrets",
  "SourceConfiguration": {
    "ImageRepository": {
      "ImageIdentifier": "<image-identifier>",
      "ImageConfiguration": {
        "Port": "<port>",
        "RuntimeEnvironmentSecrets": {

 "Credential1":"arn:aws:secretsmanager:<region>:<aws_account_id>:secret:XXXXXXXXXXXX",
```

```
          "Credential2":"arn:aws:ssm:<region>:<aws_account_id>:parameter/
<parameter-name>"
        }
      },
      "ImageRepositoryType": "ECR_PUBLIC"
    }
  },
  "InstanceConfiguration": {
    "Cpu": "1 vCPU",
    "Memory": "3 GB",
    "InstanceRoleArn": "<instance-role-arn>"
  }
}
```

Example Layanan App Runner berbasis gambar - Permintaan

```
aws apprunner create-service \
--cli-input-json file://input.json
```

Example Layanan App Runner berbasis gambar - Respons

```
{
...
      "ImageRepository": {
         "ImageIdentifier":"<image-identifier>",
         "ImageConfiguration":{
            "Port": "<port>",
            "RuntimeEnvironmentSecrets":{
               "Credential1":
 "arn:aws:secretsmanager:<region>:<aws_account_id>:secret:XXXXXXXXXXXX",
               "Credential2": "arn:aws:ssm:<region>:<aws_account_id>:parameter/
<parameter-name>"
            },
            "ImageRepositoryType":"ECR"
      }
   },
   "InstanceConfiguration": {
        "CPU": "1 vCPU",
        "Memory": "3 GB",
        "InstanceRoleArn: "<instance-role-arn>"
   }
...
```

```
}
```

Example Masukan file.json untuk layanan App Runner berbasis kode

```
{
  "ServiceName": "example-secrets",
  "SourceConfiguration": {
    "AuthenticationConfiguration": {
      "ConnectionArn": "arn:aws:apprunner:us-east-1:123456789012:connection/my-
github-connection/XXXXXXXXXX"
    },
    "AutoDeploymentsEnabled": false,
    "CodeRepository": {
      "RepositoryUrl": "<repository-url>",
      "SourceCodeVersion": {
        "Type": "BRANCH",
        "Value": "main"
      },
      "CodeConfiguration": {
        "ConfigurationSource": "API",
        "CodeConfigurationValues": {
          "Runtime": "<runtime>",
          "BuildCommand": "<build-command>",
          "StartCommand": "<start-command>",
          "Port": "<port>",
          "RuntimeEnvironmentSecrets": {

 "Credential1":"arn:aws:secretsmanager:<region>:<aws_account_id>:secret:XXXXXXXXXXXX",
            "Credential2":"arn:aws:ssm:<region>:<aws_account_id>:parameter/
<parameter-name>"
          }
        }
      }
    }
  },
  "InstanceConfiguration": {
    "Cpu": "1 vCPU",
    "Memory": "3 GB",
    "InstanceRoleArn": "<instance-role-arn>"
  }
}
```

Example Layanan Pelari Aplikasi berbasis kode - Permintaan

```
aws apprunner create-service \
--cli-input-json file://input.json
```

Example Layanan Pelari Aplikasi berbasis kode - Respons

```
{
...
   "SourceConfiguration":{
      "CodeRepository":{
         "RepositoryUrl":"<repository-url>",
         "SourceCodeVersion":{
            "Type":"Branch",
            "Value":"main"
         },
         "CodeConfiguration":{
            "ConfigurationSource":"API",
            "CodeConfigurationValues":{
               "Runtime":"<runtime>",
               "BuildCommand":"<build-command>",
               "StartCommand":"<start-command>",
               "Port":"<port>",
               "RuntimeEnvironmentSecrets":{
                  "Credential1" :
 "arn:aws:secretsmanager:<region>:<aws_account_id>:secret:XXXXXXXX",
                  "Credential2" : "arn:aws:ssm:<region>:<aws_account_id>:parameter/
<parameter-name>"
               }
            }
         }
      },
      "InstanceConfiguration": {
        "CPU": "1 vCPU",
        "Memory": "3 GB",
        "InstanceRoleArn: "<instance-role-arn>"
   }
...
}
```

3. `apprunner.yaml`Model diperbarui untuk mencerminkan rahasia yang ditambahkan.

Berikut ini adalah contoh `apprunner.yaml` model yang diperbarui.

Example **`apprunner.yaml`**

```
version: 1.0
runtime: python3
build:
  commands:
    build:
      - python -m pip install flask
run:
  command: python app.py
  network:
    port: 8080
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
  secrets:
    - name: my-secret
      value-from:
 "arn:aws:secretsmanager:<region>:<aws_account_id>:secret:XXXXXXXXXXXX"
    - name: my-parameter
      value-from: "arn:aws:ssm:<region>:<aws_account_id>:parameter/<parameter-
name>"
    - name: my-parameter-only-name
      value-from: "parameter-name"
```

Jaringan dengan App Runner

Bab ini menjelaskan konfigurasi jaringan untuk AWS App Runner layanan Anda.

Dari Bab ini Anda akan mempelajari hal-hal berikut:

- Cara mengkonfigurasi lalu lintas masuk Anda untuk titik akhir pribadi dan publik. Untuk informasi selengkapnya, lihat Menyiapkan konfigurasi jaringan untuk lalu lintas masuk.
- Cara mengonfigurasi lalu lintas keluar Anda untuk mengakses ke aplikasi lain yang berjalan di VPC Amazon. Untuk informasi selengkapnya, lihat Mengaktifkan akses VPC untuk lalu lintas keluar.

> **Note**
>
> App Runner saat ini mendukung tipe alamat dual-stack (IPv4 dan IPv6) hanya untuk lalu lintas masuk publik. Untuk lalu lintas keluar dan lalu lintas masuk pribadi hanya IPv4 didukung.

Topik

- Terminologi
- Menyiapkan konfigurasi jaringan untuk lalu lintas masuk
- Mengaktifkan akses VPC untuk lalu lintas keluar

Terminologi

Untuk mengetahui cara menyesuaikan lalu lintas jaringan Anda agar sesuai dengan kebutuhan Anda, mari kita pahami istilah-istilah berikut yang digunakan dalam Bab ini.

Ketentuan Umum

Untuk mengetahui apa yang diperlukan untuk mengaitkan dengan Amazon Virtual Private Cloud (VPC), mari kita pahami istilah-istilah berikut:

- VPC: VPC Amazon adalah jaringan virtual yang terisolasi secara logis yang memberi Anda kontrol penuh atas lingkungan jaringan virtual Anda, termasuk penempatan sumber daya, konektivitas,

dan keamanan. Ini adalah jaringan virtual yang sangat mirip dengan jaringan tradisional yang akan Anda operasikan di pusat data Anda sendiri.

- Titik akhir antarmuka VPC: Titik akhir antarmuka VPC, sumber daya AWS PrivateLink , menghubungkan VPC ke layanan titik akhir. Buat titik akhir antarmuka VPC untuk mengirim lalu lintas ke layanan endpoint yang menggunakan Network Load Balancer untuk mendistribusikan lalu lintas. Lalu lintas yang ditujukan untuk layanan titik akhir diselesaikan menggunakan DNS.
- Wilayah: Setiap Wilayah adalah area geografis terpisah tempat Anda dapat meng-host layanan App Runner.
- Availability Zone: Availability Zone adalah lokasi terisolasi di dalam suatu AWS Wilayah. Ini adalah satu atau lebih pusat data diskrit dengan daya redundan, jaringan, dan konektivitas. Availability Zones membantu Anda membuat aplikasi produksi sangat tersedia, toleran terhadap kesalahan, dan skalabel.
- Subnet: Subnet adalah berbagai alamat IP di VPC Anda. Subnet harus berada di Availability Zone tunggal. Anda dapat meluncurkan sumber AWS daya ke subnet tertentu. Ggunakan subnet publik untuk sumber daya yang harus terhubung ke internet, dan subnet privat untuk sumber daya yang tidak perlu terhubung ke internet.
- Grup keamanan: Grup keamanan mengontrol lalu lintas yang diizinkan untuk mencapai dan meninggalkan sumber daya yang terkait dengannya. Grup keamanan menyediakan lapisan keamanan tambahan untuk melindungi AWS sumber daya di setiap subnet, memberi Anda kontrol lebih besar atas lalu lintas jaringan Anda. Saat Anda membuat VPC, VPC dilengkapi dengan grup keamanan default. Anda dapat membuat grup keamanan tambahan untuk setiap VPC. Anda dapat mengaitkan grup keamanan hanya dengan sumber daya dalam VPC tempat ia dibuat.
- Dual stack: Dual stack adalah jenis alamat yang mendukung lalu lintas jaringan dari keduanya IPv4 dan IPv6 titik akhir.

Istilah khusus untuk mengonfigurasi lalu lintas keluar

Konektor VPC

Konektor VPC adalah sumber daya App Runner yang memungkinkan layanan App Runner mengakses aplikasi yang berjalan di VPC Amazon pribadi.

Ketentuan khusus untuk mengonfigurasi lalu lintas masuk

Untuk mengetahui bagaimana Anda dapat membuat layanan Anda dapat diakses secara pribadi hanya dari dalam VPC Amazon, mari kita pahami istilah-istilah berikut:

- Koneksi Ingress VPC: VPC Ingress Connection adalah sumber daya App Runner yang menyediakan titik akhir App Runner untuk lalu lintas masuk. App Runner menetapkan sumber daya VPC Ingress Connection di belakang layar saat Anda memilih titik akhir Pribadi di konsol App Runner untuk lalu lintas masuk Anda. Sumber daya VPC Ingress Connection menghubungkan layanan App Runner Anda ke titik akhir antarmuka VPC VPC Amazon.

> **Note**
>
> Jika Anda menggunakan App Runner API, sumber daya VPC Ingress Connection tidak dibuat secara otomatis.

- Titik akhir pribadi: Titik akhir pribadi adalah opsi konsol App Runner yang Anda pilih untuk mengonfigurasi lalu lintas jaringan masuk agar dapat diakses hanya dari dalam VPC Amazon.

Menyiapkan konfigurasi jaringan untuk lalu lintas masuk

Anda dapat mengonfigurasi layanan Anda untuk menerima lalu lintas masuk dari titik akhir pribadi atau publik.

Public Endpoint adalah konfigurasi default. Ini membuka layanan Anda untuk setiap lalu lintas masuk dari internet publik. Ini juga memberi Anda fleksibilitas untuk memilih antara Internet Protocol versi 4 (IPv4) atau dual-stack (IPv4 dan IPv6) jenis alamat untuk layanan Anda.

Titik akhir Pribadi hanya mengizinkan lalu lintas dari VPC Amazon untuk mengakses layanan App Runner Anda. Ini dicapai dengan menyiapkan titik akhir antarmuka VPC, AWS PrivateLink sumber daya, untuk layanan App Runner Anda. Dengan demikian, membuat koneksi pribadi antara Amazon VPC dan layanan App Runner Anda.

> **Note**
>
> App Runner saat ini mendukung tipe alamat dual-stack (IPv4 dan IPv6) hanya untuk titik akhir Publik. Untuk titik akhir Privat, hanya IPv4 didukung.

Berikut ini adalah topik yang dibahas sebagai bagian dari pengaturan konfigurasi jaringan Anda untuk lalu lintas masuk:

- Cara mengonfigurasi lalu lintas masuk Anda untuk membuat layanan Anda tersedia secara pribadi hanya dari dalam VPC Amazon. Untuk informasi selengkapnya, lihat Mengaktifkan titik akhir Pribadi untuk lalu lintas masuk.
- Cara mengkonfigurasi layanan Anda untuk menerima lalu lintas internet dari jenis alamat dual-stack. Untuk informasi selengkapnya, lihat Mengaktifkan tumpukan ganda untuk lalu lintas masuk publik.

Header

Dengan App Runner Anda dapat mengakses sumber asli IPv4 dan IPv6 alamat lalu lintas yang memasuki aplikasi Anda. Alamat IP sumber asli dipertahankan dengan menetapkan header `X-Forwarded-For` permintaan kepada mereka. Ini memungkinkan aplikasi Anda untuk mengambil alamat IP sumber asli bila diperlukan.

> **Note**
>
> Jika layanan Anda dikonfigurasi untuk menggunakan titik akhir pribadi, header `X-Forwarded-For` permintaan tidak dapat digunakan untuk mengakses alamat IP sumber asli. Jika digunakan, ia mengambil nilai palsu.

Mengaktifkan titik akhir Pribadi untuk lalu lintas masuk

Secara default ketika Anda membuat AWS App Runner layanan, layanan dapat diakses melalui internet. Namun, Anda juga dapat menjadikan layanan App Runner pribadi dan hanya dapat diakses dari dalam Amazon Virtual Private Cloud (Amazon VPC).

Dengan layanan pribadi App Runner Anda, Anda memiliki kontrol penuh atas lalu lintas masuk, menambahkan lapisan keamanan tambahan. Ini sangat membantu dalam berbagai kasus penggunaan, termasuk menjalankan internal APIs, aplikasi web perusahaan, atau aplikasi yang masih dalam pengembangan yang membutuhkan tingkat privasi dan keamanan yang lebih tinggi, atau memiliki kebutuhan untuk memenuhi persyaratan kepatuhan tertentu.

> **Note**
>
> Jika aplikasi App Runner Anda memerlukan aturan kontrol lalu lintas masuk IP/CIDR sumber, Anda harus menggunakan aturan grup keamanan untuk titik akhir pribadi, bukan web WAF. ACLs Ini karena saat ini kami tidak mendukung penerusan data IP sumber permintaan ke

layanan pribadi App Runner yang terkait dengan WAF. Akibatnya, aturan IP sumber untuk layanan pribadi App Runner yang terkait dengan web WAF ACLs tidak mematuhi aturan berbasis IP.
Untuk mempelajari selengkapnya tentang grup keamanan dan keamanan infrastruktur, termasuk praktik terbaik, lihat topik berikut di Panduan Pengguna Amazon VPC: Kontrol lalu lintas jaringan dan Kontrol lalu lintas ke sumber daya AWS Anda menggunakan grup keamanan.

Jika layanan App Runner bersifat pribadi, Anda dapat mengakses layanan Anda dari dalam VPC Amazon. Gateway internet, perangkat NAT, atau koneksi VPN tidak diperlukan.

> **Note**
>
> App Runner saat ini mendukung tipe alamat dual-stack (IPv4 dan IPv6) hanya untuk lalu lintas masuk publik. Untuk lalu lintas keluar dan lalu lintas masuk pribadi hanya IPv4 didukung.

Pertimbangan

- Sebelum menyiapkan titik akhir antarmuka VPC untuk App Runner, tinjau Pertimbangan dalam Panduan.AWS PrivateLink
- Kebijakan titik akhir VPC tidak didukung untuk App Runner. Secara default, akses penuh ke App Runner diizinkan melalui titik akhir antarmuka VPC. Atau, Anda dapat mengaitkan grup keamanan dengan antarmuka jaringan titik akhir untuk mengontrol lalu lintas ke App Runner melalui titik akhir antarmuka VPC.
- Jika aplikasi App Runner Anda memerlukan aturan kontrol lalu lintas masuk IP/CIDR sumber, Anda harus menggunakan aturan grup keamanan untuk titik akhir pribadi, bukan web WAF. ACLs Ini karena saat ini kami tidak mendukung penerusan data IP sumber permintaan ke layanan pribadi App Runner yang terkait dengan WAF. Akibatnya, aturan IP sumber untuk layanan pribadi App Runner yang terkait dengan web WAF ACLs tidak mematuhi aturan berbasis IP.
- Setelah Anda mengaktifkan titik akhir Pribadi, layanan Anda hanya dapat diakses dari VPC Anda, dan tidak dapat diakses dari internet.
- Untuk ketersediaan yang lebih tinggi, Anda disarankan untuk memilih setidaknya dua subnet di Availability Zone yang berbeda untuk titik akhir antarmuka VPC. Kami tidak menyarankan hanya menggunakan satu subnet.

- Anda dapat menggunakan titik akhir antarmuka VPC yang sama untuk mengakses beberapa layanan App Runner di VPC.

Untuk informasi tentang istilah yang digunakan di bagian ini, lihat Terminologi.

Izin

Berikut ini adalah daftar izin yang diperlukan untuk mengaktifkan titik akhir Pribadi:

- EC2: CreateTags
- EC2: CreateVpcEndpoint
- EC2: ModifyVpcEndpoint
- EC2: DeleteVpcEndpoints
- EC2: DescribeSubnets
- EC2: DescribeVpcEndpoints
- EC2: DescribeVpcs

Titik akhir antarmuka VPC

Titik akhir antarmuka VPC adalah AWS PrivateLinksumber daya yang menghubungkan VPC Amazon ke layanan titik akhir. Anda dapat menentukan VPC Amazon mana yang Anda inginkan agar layanan App Runner dapat diakses dengan melewati titik akhir antarmuka VPC. Untuk membuat antarmuka VPC endpoint tentukan hal berikut:

- VPC Amazon untuk mengaktifkan konektivitas.
- Tambahkan grup Keamanan. Secara default, grup keamanan ditetapkan ke titik akhir antarmuka VPC. Anda dapat memilih untuk mengaitkan grup keamanan khusus untuk membawa kontrol lebih lanjut ke lalu lintas jaringan yang masuk.
- Tambahkan subnet. Untuk memastikan ketersediaan yang lebih tinggi, disarankan untuk memilih setidaknya dua subnet untuk setiap Availability Zone tempat Anda akan mengakses layanan App Runner. Endpoint antarmuka jaringan dibuat di setiap subnet yang Anda aktifkan untuk titik akhir antarmuka VPC. Ini adalah antarmuka jaringan yang dikelola pemohon yang berfungsi sebagai titik masuk untuk lalu lintas yang ditujukan untuk App Runner. Antarmuka jaringan yang dikelola pemohon adalah antarmuka jaringan yang dibuat oleh AWS layanan di VPC Anda atas nama Anda.

- Jika Anda menggunakan API, tambahkan titik akhir antarmuka VPC App Runner. `Servicename` Misalnya,

```
com.amazonaws.region.apprunner.requests
```

Anda dapat membuat titik akhir antarmuka VPC menggunakan salah satu layanan berikut: AWS

- Konsol Pelari Aplikasi. Untuk informasi selengkapnya, lihat Mengelola titik akhir Pribadi.
- Konsol Amazon VPC atau API, dan AWS Command Line Interface ()AWS CLI. Untuk informasi selengkapnya, lihat Akses Layanan AWS melalui AWS PrivateLink di AWS PrivateLink Panduan.

> **Note**
>
> Anda dikenakan biaya untuk setiap titik akhir antarmuka VPC yang Anda gunakan berdasarkan Harga.AWS PrivateLink Oleh karena itu, untuk efisiensi biaya yang lebih baik, Anda dapat menggunakan titik akhir antarmuka VPC yang sama untuk mengakses beberapa layanan App Runner dalam VPC. Namun, untuk isolasi yang lebih baik, pertimbangkan untuk mengaitkan titik akhir antarmuka VPC yang berbeda untuk setiap layanan App Runner Anda.

Koneksi Masuk VPC

Koneksi Ingress VPC adalah sumber daya App Runner yang menentukan titik akhir App Runner untuk lalu lintas masuk. App Runner menetapkan sumber daya VPC Ingress Connection di belakang layar saat Anda memilih titik akhir Pribadi di konsol App Runner untuk lalu lintas masuk Anda. Pilih opsi ini untuk hanya mengizinkan lalu lintas dari VPC Amazon untuk mengakses layanan App Runner Anda. Sumber daya VPC Ingress Connection menghubungkan layanan App Runner Anda ke titik akhir antarmuka VPC VPC Amazon. Anda dapat membuat sumber daya VPC Ingress Connection hanya jika Anda menggunakan operasi API untuk mengonfigurasi pengaturan jaringan untuk lalu lintas masuk. Untuk informasi selengkapnya cara membuat resource VPC Ingress Connection, lihat CreateVpcIngressConnectiondi Referensi API.AWS App Runner

> **Note**
>
> Satu sumber daya VPC Ingress Connection dari App Runner dapat terhubung ke satu titik akhir antarmuka VPC VPC Amazon. Selain itu, Anda hanya dapat membuat satu sumber daya VPC Ingress Connection untuk setiap layanan App Runner.

Titik akhir pribadi

Titik akhir pribadi adalah opsi konsol App Runner yang dapat Anda pilih jika Anda hanya ingin menerima lalu lintas masuk dari VPC Amazon. Memilih opsi titik akhir Pribadi di konsol App Runner memberi Anda opsi untuk menghubungkan layanan Anda ke VPC dengan mengonfigurasi titik akhir antarmuka VPC-nya. Di belakang layar, App Runner menetapkan sumber daya VPC Ingress Connection ke titik akhir antarmuka VPC yang Anda konfigurasikan.

> **Note**
>
> Hanya lalu lintas IPv4 jaringan yang didukung untuk titik akhir Pribadi.

Ringkasan

Jadikan layanan Anda pribadi dengan hanya mengizinkan lalu lintas dari VPC Amazon untuk mengakses layanan App Runner Anda. Untuk mencapai hal ini, Anda membuat titik akhir antarmuka VPC untuk VPC Amazon yang dipilih menggunakan App Runner atau Amazon VPC. Di konsol App Runner, Anda membuat titik akhir antarmuka VPC saat mengaktifkan titik akhir Pribadi untuk lalu lintas Masuk. App Runner kemudian secara otomatis membuat resource VPC Ingress Connection dan terhubung ke titik akhir antarmuka VPC dan layanan App Runner Anda. Ini menciptakan koneksi layanan pribadi yang memastikan bahwa hanya lalu lintas dari VPC yang dipilih yang dapat mengakses layanan App Runner Anda.

Mengelola titik akhir Pribadi

Mengelola titik akhir Privat untuk lalu lintas masuk menggunakan salah satu metode berikut:

- the section called “Konsol Pelari Aplikasi”
- the section called “API Pelari Aplikasi atau AWS CLI”

> **Note**
>
> Jika aplikasi App Runner Anda memerlukan aturan kontrol lalu lintas masuk IP/CIDR sumber, Anda harus menggunakan aturan grup keamanan untuk titik akhir pribadi, bukan web WAF. ACLs Ini karena saat ini kami tidak mendukung penerusan data IP sumber permintaan ke layanan pribadi App Runner yang terkait dengan WAF. Akibatnya, aturan IP sumber untuk layanan pribadi App Runner yang terkait dengan web WAF ACLs tidak mematuhi aturan berbasis IP.
> Untuk mempelajari selengkapnya tentang grup keamanan dan keamanan infrastruktur, termasuk praktik terbaik, lihat topik berikut di Panduan Pengguna Amazon VPC: Kontrol lalu lintas jaringan dan Kontrol lalu lintas ke sumber daya AWS Anda menggunakan grup keamanan.

Konsol Pelari Aplikasi

Saat membuat layanan menggunakan konsol App Runner, atau saat memperbarui konfigurasinya nanti, Anda dapat memilih untuk mengonfigurasi lalu lintas yang masuk.

Untuk mengonfigurasi lalu lintas masuk Anda, pilih salah satu dari berikut ini.

- Titik akhir publik: Untuk membuat layanan Anda dapat diakses oleh semua layanan melalui internet. Secara default, titik akhir Publik dipilih.
- Titik akhir pribadi: Agar layanan App Runner dapat diakses hanya dari dalam VPC Amazon.

> **Note**
>
> Saat ini, App Runner IPv6 hanya mendukung titik akhir publik. IPv6 endpoint tidak didukung untuk layanan App Runner yang dihosting di Amazon Virtual Private Cloud (Amazon VPC). Jika Anda memperbarui layanan yang menggunakan titik akhir publik tumpukan ganda ke titik akhir pribadi, layanan App Runner Anda akan secara default mendukung lalu lintas dari titik IPv4 akhir saja dan gagal menerima lalu lintas dari titik akhir. IPv6

Aktifkan titik akhir Pribadi

Aktifkan titik akhir Pribadi dengan mengaitkannya dengan titik akhir antarmuka VPC VPC Amazon yang ingin Anda akses. Anda dapat membuat titik akhir antarmuka VPC baru atau memilih yang sudah ada.

Untuk membuat titik akhir antarmuka VPC

1. Buka konsol App Runner, dan di daftar Wilayah, pilih Wilayah AWS.
2. Buka bagian Jaringan di bawah Konfigurasi layanan.
3. Pilih titik akhir pribadi, untuk lalu lintas jaringan masuk. Opsi untuk terhubung ke VCP menggunakan titik akhir antarmuka VPC terbuka.
4. Pilih Buat titik akhir baru. Kotak dialog titik akhir antarmuka VPC Create new terbuka.
5. Masukkan nama untuk titik akhir antarmuka VPC.
6. Pilih titik akhir antarmuka VPC yang diperlukan dari daftar drop-down yang tersedia.
7. Pilih grup keamanan dari daftar drop-down. Menambahkan grup keamanan menyediakan lapisan keamanan tambahan ke titik akhir antarmuka VPC. Disarankan untuk memilih dua atau lebih kelompok keamanan. Jika Anda tidak memilih grup keamanan, App Runner menetapkan grup keamanan default ke titik akhir antarmuka VPC. Pastikan aturan grup keamanan tidak memblokir resource yang ingin berkomunikasi dengan layanan App Runner Anda. Aturan grup keamanan harus mengizinkan sumber daya yang akan berinteraksi dengan layanan App Runner Anda.

   > **Note**
   >
   > Jika aplikasi App Runner Anda memerlukan aturan kontrol lalu lintas masuk IP/CIDR sumber, Anda harus menggunakan aturan grup keamanan untuk titik akhir pribadi, bukan web WAF. ACLs Ini karena saat ini kami tidak mendukung penerusan data IP sumber permintaan ke layanan pribadi App Runner yang terkait dengan WAF. Akibatnya, aturan IP sumber untuk layanan pribadi App Runner yang terkait dengan web WAF ACLs tidak mematuhi aturan berbasis IP.
   > Untuk mempelajari selengkapnya tentang grup keamanan dan keamanan infrastruktur, termasuk praktik terbaik, lihat topik berikut di Panduan Pengguna Amazon VPC: Kontrol lalu lintas jaringan dan Kontrol lalu lintas ke sumber daya AWS Anda menggunakan grup keamanan.

8. Pilih subnet yang diperlukan dari daftar drop-down. Disarankan untuk memilih setidaknya dua subnet untuk setiap Availability Zone dari mana Anda akan mengakses layanan App Runner.

9. (Opsional) Pilih Tambahkan tag baru dan masukkan kunci tag dan nilai tag.
10. Pilih Buat. Halaman Konfigurasi layanan terbuka yang menampilkan pesan keberhasilan pembuatan titik akhir antarmuka VPC di bilah atas.

Untuk memilih titik akhir antarmuka VPC yang ada

1. Buka konsol App Runner, dan di daftar Wilayah, pilih Wilayah AWS.
2. Buka bagian Jaringan di bawah Konfigurasi layanan.
3. Pilih titik akhir pribadi, untuk lalu lintas jaringan masuk. Opsi untuk terhubung ke VPC menggunakan titik akhir antarmuka VPC terbuka. Daftar titik akhir antarmuka VPC yang tersedia ditampilkan.
4. Pilih titik akhir antarmuka VPC yang diperlukan yang tercantum di bawah titik akhir antarmuka VPC.
5. Pilih Berikutnya untuk membuat layanan Anda. App Runner mengaktifkan titik akhir Pribadi.

> **Note**
>
> Setelah layanan dibuat, Anda dapat memilih untuk mengedit grup Keamanan dan Subnet yang terkait dengan titik akhir antarmuka VPC, jika diperlukan.

Untuk memeriksa detail titik akhir Pribadi, buka layanan Anda dan perluas bagian Jaringan di bawah tab Konfigurasi. Ini menunjukkan detail VPC dan titik akhir antarmuka VPC yang terkait dengan titik akhir Private.

Perbarui titik akhir antarmuka VPC

Setelah layanan App Runner dibuat, Anda dapat mengedit titik akhir antarmuka VPC yang terkait dengan titik akhir Privat.

> **Note**
>
> Anda tidak dapat memperbarui nama Endpoint dan bidang VPC.

Untuk memperbarui titik akhir antarmuka VPC

1. Buka konsol App Runner, dan di daftar Wilayah, pilih Wilayah AWS.
2. Buka layanan Anda dan pilih Konfigurasi jaringan di panel kiri.
3. Pilih Lalu lintas masuk untuk melihat titik akhir antarmuka VPC yang terkait dengan layanan masing-masing.
4. Pilih titik akhir antarmuka VPC yang ingin Anda edit.
5. Pilih Edit. Kotak dialog untuk mengedit titik akhir antarmuka VPC terbuka.
6. Pilih grup Keamanan dan Subnet yang diperlukan dan klik Perbarui. Halaman yang menampilkan detail titik akhir antarmuka VPC terbuka dengan pesan pembaruan yang berhasil dari titik akhir antarmuka VPC di bilah atas.

> **Note**
>
> Jika aplikasi App Runner Anda memerlukan aturan kontrol lalu lintas masuk IP/CIDR sumber, Anda harus menggunakan aturan grup keamanan untuk titik akhir pribadi, bukan web WAF. ACLs Ini karena saat ini kami tidak mendukung penerusan data IP sumber permintaan ke layanan pribadi App Runner yang terkait dengan WAF. Akibatnya, aturan IP sumber untuk layanan pribadi App Runner yang terkait dengan web WAF ACLs tidak mematuhi aturan berbasis IP.
> Untuk mempelajari selengkapnya tentang grup keamanan dan keamanan infrastruktur, termasuk praktik terbaik, lihat topik berikut di Panduan Pengguna Amazon VPC: Kontrol lalu lintas jaringan dan Kontrol lalu lintas ke sumber daya AWS Anda menggunakan grup keamanan.

Hapus titik akhir antarmuka VPC

Jika Anda tidak ingin layanan App Runner dapat diakses secara pribadi, Anda dapat mengatur lalu lintas masuk ke Publik. Mengubah ke Publik menghapus titik akhir Pribadi, tetapi tidak menghapus titik akhir antarmuka VPC

Untuk menghapus titik akhir antarmuka VPC

1. Buka konsol App Runner, dan di daftar Wilayah, pilih Wilayah AWS.
2. Buka layanan Anda dan pilih Konfigurasi jaringan di panel kiri.

3. Pilih Lalu lintas masuk untuk melihat titik akhir antarmuka VPC yang terkait dengan layanan masing-masing.

> **Note**
>
> Sebelum menghapus titik akhir antarmuka VPC, hapus dari semua layanan yang terhubung dengan memperbarui layanan Anda.

4. Pilih Hapus.

   Jika ada layanan yang terhubung ke titik akhir antarmuka VPC, maka Anda menerima pesan titik akhir antarmuka VPC Tidak dapat menghapus. Jika tidak ada layanan yang terhubung ke titik akhir antarmuka VPC, Anda menerima pesan untuk mengonfirmasi penghapusan.

5. Pilih Hapus. Halaman Konfigurasi Jaringan terbuka untuk lalu lintas masuk dengan pesan penghapusan titik akhir antarmuka VPC yang berhasil di bilah atas.

API Pelari Aplikasi atau AWS CLI

Anda dapat menerapkan aplikasi di App Runner yang hanya dapat diakses dari dalam VPC Amazon.

Untuk informasi tentang izin yang diperlukan untuk menjadikan layanan Anda pribadi, lihatthe section called “Izin”.

> **Note**
>
> Saat ini, App Runner IPv6 hanya mendukung titik akhir publik. IPv6 endpoint tidak didukung untuk layanan App Runner yang dihosting di Amazon Virtual Private Cloud (Amazon VPC). Jika Anda memperbarui layanan yang menggunakan titik akhir publik tumpukan ganda ke titik akhir pribadi, layanan App Runner Anda akan secara default mendukung lalu lintas dari titik IPv4 akhir saja dan gagal menerima lalu lintas dari titik akhir. IPv6

Untuk membuat koneksi layanan pribadi ke Amazon VPC

1. Buat titik akhir antarmuka VPC, AWS PrivateLink sumber daya, untuk terhubung ke App Runner. Untuk melakukan ini, tentukan subnet dan grup keamanan untuk dikaitkan dengan aplikasi. Berikut ini adalah contoh pembuatan titik akhir antarmuka VPC.

> **Note**
>
> Jika aplikasi App Runner Anda memerlukan aturan kontrol lalu lintas masuk IP/CIDR sumber, Anda harus menggunakan aturan grup keamanan untuk titik akhir pribadi, bukan web WAF. ACLs Ini karena saat ini kami tidak mendukung penerusan data IP sumber permintaan ke layanan pribadi App Runner yang terkait dengan WAF. Akibatnya, aturan IP sumber untuk layanan pribadi App Runner yang terkait dengan web WAF ACLs tidak mematuhi aturan berbasis IP.
> Untuk mempelajari selengkapnya tentang grup keamanan dan keamanan infrastruktur, termasuk praktik terbaik, lihat topik berikut di Panduan Pengguna Amazon VPC: Kontrol lalu lintas jaringan dan Kontrol lalu lintas ke sumber daya AWS Anda menggunakan grup keamanan.

Example

```
aws ec2 create-vpc-endpoint
 --vpc-endpoint-type: Interface
 --service-name: com.amazonaws.us-east-1.apprunner.requests
 --subnets: subnet1, subnet2
 --security-groups: sg1
```

2. Referensikan titik akhir antarmuka VPC dengan menggunakan tindakan API CreateServiceatau UpdateServiceApp Runner melalui CLI. Konfigurasikan layanan Anda agar tidak dapat diakses secara publik. Setel `IsPubliclyAccessible` ke `False` `IngressConfiguration` anggota `NetworkConfiguration` parameter. Berikut ini adalah contoh referensi titik akhir antarmuka VPC.

   Example

```
aws apprunner create-service
 --network-configuration: ingress-configuration=<ingress_configuration>
 --service-name: com.amazonaws.us-east-1.apprunner.requests
 --source-configuration: <source_configuration>
 # Ingress Configuration
 {
 "IsPubliclyAccessible": False
 }
```

3. Panggil tindakan `create-vpc-ingress-connection` API untuk membuat resource VPC Ingress Connection untuk App Runner dan kaitkan dengan titik akhir antarmuka VPC yang Anda buat pada langkah sebelumnya. Ini mengembalikan nama domain yang digunakan untuk mengakses layanan Anda di VPC yang ditentukan. Berikut ini adalah contoh pembuatan sumber daya VPC Ingress Connection.

   Example Permintaan

   ```
   aws apprunner create-vpc-ingress-connection
    --service-arn: <apprunner_service_arn>
    --ingress-vpc-configuration: {"VpcId":<vpc_id>, "VpceId": <vpce_id>}
    --vpc-ingress-connection-name: <vic_connection_name>
   ```

   Example Respons

   ```
   {
      "VpcIngressConnectionArn": <vpc_ingress_connection_arn>,
      "VpcIngressConnectionName": <vic_connection_name>,
      "ServiceArn": <apprunner_service_arn>,
      "Status": "PENDING_CREATION",
      "AccountId": <connection_owner_id>,
      "DomainName": <domain_name_associated_with_vpce>,
      "IngressVpcConfiguration": {"VpcId":<vpc_id>, "VpceId":<vpce_id>},
      "CreatedAt": <date_created>
   }
   ```

Perbarui Koneksi Ingress VPC

Anda dapat memperbarui sumber daya VPC Ingress Connection. Koneksi Ingress VPC harus dalam salah satu status berikut untuk diperbarui:

- AVAILABLE
- FAILED_CREATION
- FAILED_UPDATE

Berikut ini adalah contoh memperbarui sumber daya VPC Ingress Connection.

Example Permintaan

```
aws apprunner update-vpc-ingress-connection
      --vpc-ingress-connection-arn: <vpc_ingress_connection_arn>
```

Example Respons

```
{
    "VpcIngressConnectionArn":  <vpc_ingress_connection_arn>,
    "VpcIngressConnectionName":  <vic_connection_name>,
    "ServiceArn":  <apprunner_service_arn>,
    "Status": "FAILED_UPDATE",
    "AccountId":  <connection_owner_id>,
    "DomainName":  <domain_name_associated_with_vpce>,
    "IngressVpcConfiguration": {"VpcId":<vpc_id>, "VpceId":<vpce_id>},
    "CreatedAt":  <date_created>
}
```

Hapus Koneksi Ingress VPC

Anda dapat menghapus sumber daya VPC Ingress Connection jika Anda tidak lagi memerlukan koneksi pribadi ke VPC Amazon.

Koneksi Ingress VPC harus dalam salah satu status berikut untuk dihapus:

- AVAILABLE
- PENCIPTAAN GAGAL
- PEMBARUAN GAGAL
- PENGHAPUSAN GAGAL

Berikut ini adalah contoh menghapus Koneksi Ingress VPC

Example Permintaan

```
aws apprunner delete-vpc-ingress-connection
      --vpc-ingress-connection-arn: <vpc_ingress_connection_arn>
```

Example Respons

```
{
```

```
    "VpcIngressConnectionArn": <vpc_ingress_connection_arn>,
    "VpcIngressConnectionName": <vic_connection_name>,
    "ServiceArn": <apprunner_service_arn>,
    "Status": "PENDING_DELETION",
    "AccountId": <connection_owner_id>,
    "DomainName": <domain_name_associated_with_vpce>,
    "IngressVpcConfiguration": {"VpcId":<vpc_id>, "VpceId":<vpce_id>},
    "CreatedAt": <date_created>,
    "DeletedAt": <date_deleted>
}
```

Gunakan tindakan App Runner API berikut untuk mengelola lalu lintas masuk pribadi untuk layanan Anda.

- CreateVpcIngressConnection— Buat sumber daya Koneksi Ingress VPC baru. App Runner memerlukan sumber daya ini saat Anda ingin mengaitkan layanan App Runner ke titik akhir Amazon VPC.
- ListVpcIngressConnections— Kembalikan daftar titik akhir AWS App Runner VPC Ingress Connection yang terkait dengan akun Anda. AWS
- DescribeVpcIngressConnection— Kembalikan deskripsi lengkap sumber daya AWS App Runner VPC Ingress Connection.
- UpdateVpcIngressConnection— Perbarui sumber daya AWS App Runner Koneksi Ingress VPC.
- DeleteVpcIngressConnection— Hapus sumber daya Koneksi Ingress VPC Pelari Aplikasi yang terkait dengan layanan App Runner.

Untuk informasi selengkapnya tentang penggunaan App Runner API, lihat Panduan Referensi API Pelari Aplikasi.

Mengaktifkan lalu IPv6 lintas masuk publik

Jika Anda ingin layanan Anda menerima lalu lintas jaringan masuk dari IPv6 alamat, atau dari keduanya IPv4 dan IPv6 alamat, pilih jenis alamat Dual-stack untuk titik akhir publik. Saat Anda membuat aplikasi baru, Anda dapat menemukan pengaturan ini di bawah bagian Configure service > Networking. Untuk informasi selengkapnya tentang cara mengaktifkan IPv6 menggunakan konsol App Runner atau App Runner API, lihat. the section called “Kelola tumpukan ganda untuk titik akhir publik”

Untuk informasi lebih lanjut tentang mengadopsi IPv6 AWS, lihat IPv6 di AWS.

App Runner mendukung tumpukan ganda hanya untuk titik akhir layanan App Runner publik. Untuk semua layanan pribadi App Runner, hanya IPv4 didukung.

> **Note**
>
> Ketika Anda memiliki jenis alamat IP yang disetel ke Dual-stack dan Anda mengubah konfigurasi jaringan Anda dari titik akhir publik ke pribadi, App Runner akan secara otomatis mengubah jenis alamat Anda menjadi. IPv4 Ini karena App Runner IPv6 hanya mendukung titik akhir publik.

Pelajari informasi latar belakang tentang IPv4 vs IPv6

Lapisan IPv4 jaringan, yang biasa digunakan untuk merutekan lalu lintas jaringan di internet, menggunakan skema alamat 32-bit. Ruang alamat ini terbatas dan dapat habis dengan sejumlah besar perangkat jaringan. Untuk alasan ini, Network Address Translation (NAT) biasanya digunakan untuk merutekan beberapa IPv4 alamat melalui satu alamat jaringan publik.

IPv6, versi Internet Protocol yang lebih baru, dibangun di atas IPv4 dan memperluas ruang alamat dengan skema pengalamatan 128-bit. Dengan IPv6, Anda dapat membangun jaringan dengan jumlah perangkat yang terhubung hampir tidak terbatas. Karena banyaknya alamat jaringan, NAT tidak diperlukan oleh IPv6.

IPv4 dan IPv6 titik akhir tidak kompatibel satu sama lain karena IPv4 titik akhir tidak dapat menerima IPv6 lalu lintas masuk dan sebaliknya. Dual stack memberikan solusi yang nyaman, di mana keduanya IPv4 dan lalu lintas IPv6 jaringan dapat didukung secara bersamaan.

Mengelola tumpukan ganda untuk lalu lintas masuk publik

Mengelola jenis alamat dual-stack untuk lalu lintas masuk publik menggunakan salah satu metode berikut:

- the section called “Konsol Pelari Aplikasi”
- the section called “API Pelari Aplikasi atau AWS CLI”

Konsol Pelari Aplikasi

Anda dapat memilih jenis alamat dual-stack untuk lalu lintas internet yang masuk, saat membuat layanan menggunakan konsol App Runner, atau saat memperbarui konfigurasinya nanti.

Untuk mengaktifkan tipe alamat dual-stack

1. Saat membuat atau memperbarui layanan, perluas bagian Jaringan di bawah Konfigurasi layanan.
2. Pilih titik akhir Publik, untuk lalu lintas jaringan masuk. Opsi jenis alamat IP titik akhir publik terbuka.
3. Perluas jenis alamat IP titik akhir Publik untuk melihat jenis alamat IP berikut.
   - IPv4
   - Tumpukan ganda (dan) IPv4 IPv6

   > **Note**
   >
   > Jika Anda tidak memperluas jenis alamat IP titik akhir Publik untuk membuat pilihan, maka App Runner menetapkan IPv4 sebagai konfigurasi default.

4. Pilih Dual-stack (IPv4 dan IPv6).
5. Pilih Berikutnya dan kemudian Buat & Terapkan jika Anda membuat layanan. Jika tidak, pilih Simpan perubahan jika Anda memperbarui layanan.

   Ketika layanan digunakan, aplikasi Anda mulai menerima lalu lintas jaringan dari keduanya IPv4 dan titik IPv6 akhir.

> **Note**
>
> Saat ini, App Runner IPv6 hanya mendukung titik akhir publik. IPv6 endpoint tidak didukung untuk layanan App Runner yang dihosting di Amazon Virtual Private Cloud (Amazon VPC). Jika Anda memperbarui layanan yang menggunakan titik akhir publik tumpukan ganda ke titik akhir pribadi, layanan App Runner Anda akan secara default mendukung lalu lintas dari titik IPv4 akhir saja dan gagal menerima lalu lintas dari titik akhir. IPv6

Untuk mengubah jenis alamat

1. Ikuti langkah-langkah untuk memperbarui layanan dan menavigasi ke Jaringan.

2. Arahkan ke jenis alamat IP titik akhir publik di bawah Lalu lintas jaringan masuk dan pilih jenis alamat yang diperlukan.
3. Pilih Simpan perubahan. Layanan Anda diperbarui dengan pilihan Anda.

API Pelari Aplikasi atau AWS CLI

Saat Anda memanggil tindakan CreateServiceatau UpdateServiceApp Runner API, gunakan `IpAddressType` anggota `NetworkConfiguration` parameter untuk menentukan jenis alamat. Nilai yang didukung yang dapat Anda tentukan adalah `IPv4` dan`DUAL_STACK`. Tentukan `DUAL_STACK` apakah Anda ingin layanan Anda menerima lalu lintas internet dari IPv4 dan IPv6 titik akhir. Jika Anda tidak menentukan nilai apa pun untuk`IpAddressType`, secara default IPv4 diterapkan.

Berikut ini adalah contoh untuk membuat layanan dengan dual stack sebagai alamat IP. Contoh ini memanggil `input.json` file.

Example Permintaan untuk membuat layanan dengan dukungan tumpukan ganda

```
aws apprunner create-service \
 --cli-input-json file://input.json
```

Example Isi dari **`input.json`**

```
{
  "ServiceName": "example-service",
  "SourceConfiguration": {
    "ImageRepository": {
      "ImageIdentifier": "public.ecr.aws/aws-containers/hello-app-runner:latest",
      "ImageConfiguration": {
        "Port": "8000"
      },
      "ImageRepositoryType": "ECR_PUBLIC"
    },
    "NetworkConfiguration": {
      "IpAddressType": "DUAL_STACK"
    }
  }
}
```

Example Respons

```
{
  "Service": {
    "ServiceName": "example-service",
    "ServiceId": "<service-id>",
    "ServiceArn": "arn:aws:apprunner:us-east-2:123456789012:service/example-
service/<service-id>",
    "ServiceUrl": "1234567890.us-east-2.awsapprunner.com",
    "CreatedAt": "2023-10-16T12:30:51.724000-04:00",
    "UpdatedAt": "2023-10-16T12:30:51.724000-04:00",
    "Status": "OPERATION_IN_PROGRESS",
    "SourceConfiguration": {
      "ImageRepository": {
        "ImageIdentifier": "public.ecr.aws/aws-containers/hello-app-runner:latest",
        "ImageConfiguration": {
          "Port": "8000"
        },
        "ImageRepositoryType": "ECR_PUBLIC"
      },
      "AutoDeploymentsEnabled": false
    },
    "InstanceConfiguration": {
      "Cpu": "1024",
      "Memory": "2048"
    },
    "HealthCheckConfiguration": {
      "Protocol": "TCP",
      "Path": "/",
      "Interval": 5,
      "Timeout": 2,
      "HealthyThreshold": 1,
      "UnhealthyThreshold": 5
    },
    "AutoScalingConfigurationSummary": {
      "AutoScalingConfigurationArn": "arn:aws:apprunner:us-
east-2:123456789012:autoscalingconfiguration/
DefaultConfiguration/1/00000000000000000000000000000001",
      "AutoScalingConfigurationName": "DefaultConfiguration",
      "AutoScalingConfigurationRevision": 1
    },
    "NetworkConfiguration": {
      "IpAddressType": "DUAL_STACK",
      "EgressConfiguration": {
```

```
        "EgressType": "DEFAULT"
      },
      "IngressConfiguration": {
        "IsPubliclyAccessible": true
      }
    }
  },
  "OperationId": "24bd100b1e111ae1a1f0e1115c4f11de"
}
```

> **Note**
>
> Saat ini, App Runner IPv6 hanya mendukung titik akhir publik. IPv6 endpoint tidak didukung untuk layanan App Runner yang dihosting di Amazon Virtual Private Cloud (Amazon VPC). Jika Anda memperbarui layanan yang menggunakan titik akhir publik tumpukan ganda ke titik akhir pribadi, layanan App Runner Anda akan secara default mendukung lalu lintas dari titik IPv4 akhir saja dan gagal menerima lalu lintas dari titik akhir. IPv6

Untuk informasi selengkapnya tentang parameter API, lihat NetworkConfiguration.

Mengaktifkan akses VPC untuk lalu lintas keluar

Secara default, AWS App Runner aplikasi Anda dapat mengirim pesan ke titik akhir publik. Ini termasuk solusi Anda sendiri, Layanan AWS, dan situs web publik atau layanan web lainnya. Aplikasi Anda bahkan dapat mengirim pesan ke titik akhir publik aplikasi yang berjalan di VPC dari Amazon Virtual Private Cloud (Amazon VPC). Jika Anda tidak mengonfigurasi VPC saat meluncurkan lingkungan, App Runner menggunakan VPC default, yang bersifat publik.

Anda dapat memilih untuk meluncurkan lingkungan Anda dalam VPC khusus untuk menyesuaikan pengaturan jaringan dan keamanan untuk lalu lintas keluar. Anda dapat mengaktifkan AWS App Runner layanan Anda untuk mengakses aplikasi yang berjalan di VPC pribadi dari Amazon Virtual Private Cloud (Amazon VPC). Setelah Anda melakukan ini, aplikasi Anda dapat terhubung dengan dan mengirim pesan ke aplikasi lain yang di-host di Amazon Virtual Private Cloud (Amazon VPC). Contohnya adalah database Amazon RDS, Amazon ElastiCache, dan layanan pribadi lainnya yang di-host di VPC pribadi.

Konektor VPC

Anda dapat mengaitkan layanan Anda dengan VPC dengan membuat titik akhir VPC dari konsol App Runner, yang disebut Konektor VPC. Untuk membuat Konektor VPC, tentukan VPC, satu atau lebih subnet, dan opsional satu atau lebih grup keamanan. Setelah mengonfigurasi Konektor VPC, Anda dapat menggunakannya dengan satu atau beberapa layanan App Runner.

Latensi satu kali

Jika Anda mengonfigurasi layanan App Runner dengan konektor VPC khusus untuk lalu lintas keluar, layanan ini mungkin mengalami latensi startup satu kali selama dua hingga lima menit. Proses startup menunggu hingga Konektor VPC siap terhubung ke sumber daya lain sebelum menetapkan status layanan ke Running. Anda dapat mengonfigurasi layanan dengan konektor VPC khusus saat pertama kali membuatnya, atau Anda dapat melakukannya sesudahnya dengan melakukan pembaruan layanan.

Perhatikan bahwa jika Anda menggunakan kembali konfigurasi konektor VPC yang sama untuk layanan lain, tidak akan ada latensi apa pun. Konfigurasi konektor VPC didasarkan pada grup keamanan dan kombinasi subnet. Untuk konfigurasi konektor VPC tertentu, latensi hanya terjadi sekali, selama pembuatan awal VPC Connector Hyperplane ENIs (antarmuka jaringan elastis).

Lebih lanjut tentang konektor VPC Kustom dan Hyperplane AWS

Konektor VPC di App Runner didasarkan pada AWS Hyperplane, sistem jaringan Amazon internal yang berada di belakang beberapa AWS sumber daya, seperti Network Load Balancer, NAT Gateway, dan AWS. PrivateLink Teknologi AWS Hyperplane memberikan throughput tinggi dan kemampuan latensi rendah, bersama dengan tingkat berbagi yang lebih tinggi. Hyperplane ENI dibuat di subnet Anda saat Anda membuat konektor VPC dan mengaitkannya dengan layanan Anda. Konfigurasi konektor VPC didasarkan pada grup keamanan dan kombinasi subnet, dan Anda dapat mereferensikan Konektor VPC yang sama di beberapa layanan App Runner. Akibatnya, Hyperplane yang mendasarinya ENIs dibagikan di seluruh layanan App Runner Anda. Berbagi ini layak, bahkan saat Anda meningkatkan jumlah tugas yang diperlukan untuk menangani beban permintaan, dan menghasilkan pemanfaatan ruang IP yang lebih efisien di VPC Anda. Untuk informasi selengkapnya, lihat Deep Dive di AWS App Runner VPC Networking di AWS Container Blog.

Subnet

Setiap subnet berada di Availability Zone tertentu. Untuk ketersediaan tinggi, kami sarankan Anda memilih subnet di setidaknya tiga Availability Zone. Jika Wilayah memiliki kurang dari tiga Availability Zone, sebaiknya pilih subnet di semua Availability Zone yang didukung.

Saat memilih subnet untuk VPC Anda, pastikan Anda memilih subnet pribadi, bukan subnet publik. Ini karena, ketika Anda membuat Konektor VPC, layanan App Runner membuat ENI Hyperplane di setiap subnet. Setiap Hyperplane ENI diberi alamat IP pribadi saja dan ditandai dengan tag kunci. AWSAppRunnerManaged Jika Anda memilih subnet publik, kesalahan akan terjadi saat menjalankan layanan App Runner Anda. Namun, jika layanan Anda perlu mengakses beberapa layanan yang ada di internet atau publik lainnya Layanan AWS, lihatthe section called “Pertimbangan saat memilih subnet ”.

Pertimbangan saat memilih subnet

- Ketika Anda menghubungkan layanan Anda ke VPC, lalu lintas keluar tidak memiliki akses ke internet publik. Semua lalu lintas keluar dari aplikasi Anda diarahkan melalui VPC tempat layanan Anda terhubung. Semua aturan jaringan untuk VPC berlaku untuk lalu lintas keluar aplikasi Anda. Ini berarti bahwa layanan Anda tidak dapat mengakses internet publik dan AWS APIs. Untuk mendapatkan akses, lakukan salah satu hal berikut:
  - Hubungkan subnet ke internet melalui NAT Gateway.
  - Siapkan titik akhir VPC untuk Layanan AWS yang ingin Anda akses. Layanan Anda tetap berada dalam VPC Amazon dengan menggunakan. AWS PrivateLink
- Beberapa Availability Zone di beberapa Wilayah AWS tidak mendukung subnet yang dapat digunakan dengan layanan App Runner. Jika Anda memilih subnet di Availability Zone ini, layanan Anda gagal dibuat atau diperbarui. Untuk situasi ini, App Runner menyediakan pesan kesalahan mendetail yang menunjuk ke subnet dan Availability Zone yang tidak didukung. Ketika ini terjadi, pecahkan masalah dengan menghapus subnet yang tidak didukung dari permintaan Anda, lalu coba lagi.

Grup keamanan

Anda dapat menentukan grup keamanan yang digunakan App Runner untuk mengakses AWS subnet yang ditentukan secara opsional. Jika Anda tidak menentukan grup keamanan, App Runner menggunakan grup keamanan default VPC. Grup keamanan default memungkinkan semua lalu lintas keluar.

Menambahkan grup keamanan memberikan lapisan keamanan tambahan ke Konektor VCP, memberi Anda kontrol lebih besar atas lalu lintas jaringan. Konektor VPC hanya digunakan untuk komunikasi keluar dari aplikasi Anda. Anda menggunakan aturan keluar untuk memungkinkan komunikasi ke titik akhir tujuan yang diinginkan. Anda juga harus memastikan bahwa setiap grup keamanan yang terkait dengan sumber daya tujuan memiliki aturan masuk yang sesuai. Jika tidak, sumber daya ini tidak dapat menerima lalu lintas yang berasal dari grup keamanan Konektor VPC.

> **Note**
>
> Saat Anda mengaitkan layanan Anda dengan VPC, lalu lintas berikut tidak terpengaruh:
>
> - Lalu lintas masuk — Pesan masuk yang diterima aplikasi Anda tidak terpengaruh oleh VPC terkait. Pesan dialihkan melalui nama domain publik yang terkait dengan layanan Anda dan tidak berinteraksi dengan VPC.
> - Lalu lintas App Runner — App Runner mengelola beberapa tindakan atas nama Anda, seperti menarik kode sumber dan gambar, mendorong log, dan mengambil rahasia. Lalu lintas yang dihasilkan oleh tindakan ini tidak dirutekan melalui VPC Anda.

Untuk mengetahui lebih lanjut tentang cara AWS App Runner mengintegrasikan dengan Amazon VPC, AWS lihat Jaringan VPC Pelari Aplikasi.

> **Note**
>
> Untuk lalu lintas keluar App Runner saat ini hanya mendukung. IPv4

Kelola akses VPC

> **Note**
>
> Jika Anda membuat konektor VPC lalu lintas keluar untuk suatu layanan, proses startup layanan berikut akan mengalami latensi satu kali. Anda dapat mengatur konfigurasi ini untuk layanan baru saat Anda membuatnya, atau sesudahnya, dengan pembaruan layanan. Untuk informasi selengkapnya, lihat Latensi satu kali di bagian Networking with App Runner dari panduan ini.

Mengelola akses VPC untuk layanan App Runner Anda menggunakan salah satu metode berikut:

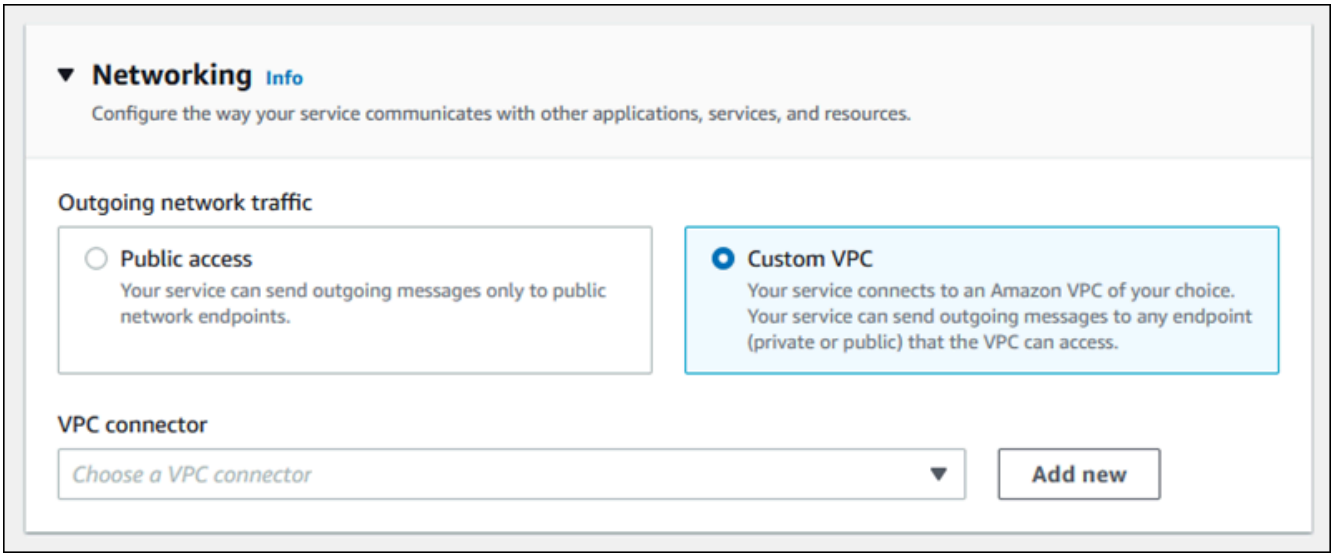
App Runner console

Saat membuat layanan menggunakan konsol App Runner, atau saat memperbarui konfigurasinya nanti, Anda dapat memilih untuk mengonfigurasi lalu lintas keluar. Cari bagian Konfigurasi jaringan di halaman konsol. Untuk lalu lintas jaringan keluar, pilih berikut ini:

- Akses publik: Untuk mengaitkan layanan Anda dengan titik akhir publik lainnya Layanan AWS.
- VPC Kustom: Untuk mengaitkan layanan Anda dengan VPC dari Amazon VPC. Aplikasi Anda dapat terhubung dengan dan mengirim pesan ke aplikasi lain yang di-host di Amazon VPC.

Untuk mengaktifkan VPC Kustom

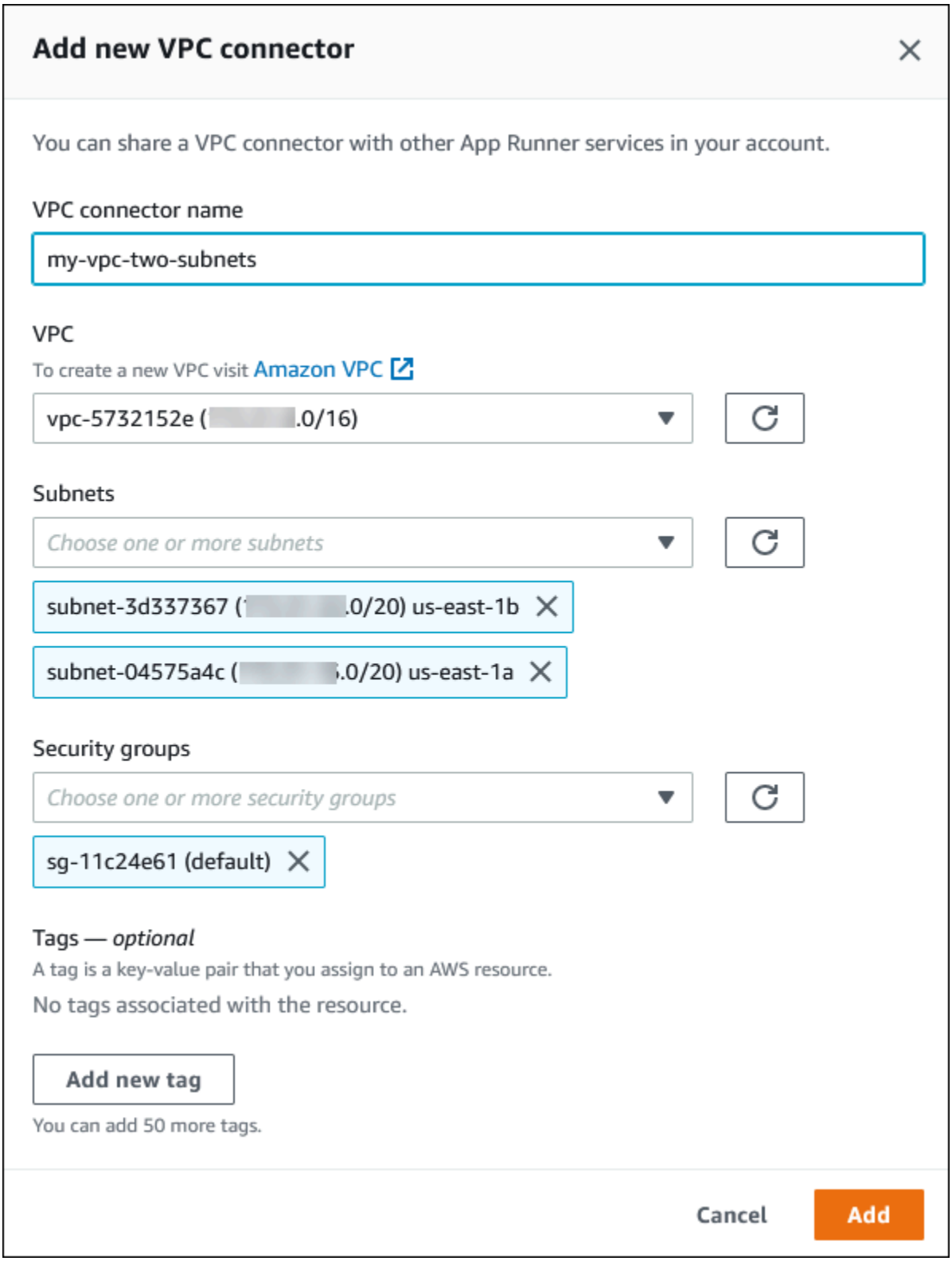
1. Buka konsol App Runner, dan di daftar Wilayah, pilih Wilayah AWS.
2. Buka bagian Jaringan di bawah Konfigurasi layanan.



3. Pilih VPC Kustom, untuk lalu lintas jaringan keluar.
4. Di panel navigasi, pilih konektor VPC.

   Jika Anda membuat konektor VPC, konsol menampilkan daftar konektor VPC di akun Anda. Anda dapat memilih konektor VPC yang ada dan memilih Berikutnya untuk meninjau konfigurasi Anda. Kemudian, pindah ke langkah terakhir. Atau, Anda dapat menambahkan konektor VPC baru menggunakan langkah-langkah berikut.
5. Pilih Tambahkan baru untuk membuat konektor VPC baru untuk layanan Anda.

Kemudian, kotak dialog Tambahkan konektor VPC baru terbuka.



6. Masukkan nama untuk konektor VPC Anda dan pilih VPC yang diperlukan dari daftar yang tersedia.
7. Untuk Subnet pilih satu subnet untuk setiap Availability Zone yang Anda rencanakan untuk mengakses layanan App Runner. Untuk ketersediaan yang lebih baik, pilih tiga subnet. Atau, jika ada kurang dari tiga subnet, pilih semua subnet yang tersedia.

   > **Note**
   >
   > Pastikan Anda menetapkan subnet pribadi ke konektor VPC. Jika Anda menetapkan subnet publik ke konektor VPC, layanan Anda gagal membuat atau memutar kembali secara otomatis selama pembaruan.

8. (Opsional) Untuk grup Keamanan, pilih grup keamanan untuk dikaitkan dengan antarmuka jaringan titik akhir.
9. (Opsional) Untuk menambahkan tanda, pilih Tambahkan tanda baru dan masukkan kunci dan nilai tanda.
10. Pilih Tambahkan.

    Detail konektor VPC yang Anda buat muncul di bawah konektor VPC.
11. Pilih Berikutnya untuk meninjau konfigurasi Anda, lalu pilih Buat dan terapkan.

    App Runner membuat sumber daya konektor VPC untuk Anda, dan kemudian mengaitkannya dengan layanan Anda. Jika layanan berhasil dibuat, konsol menampilkan dasbor layanan, dengan ikhtisar Layanan dari layanan baru.

App Runner API or AWS CLI

Saat Anda memanggil tindakan CreateServiceatau UpdateServiceApp Runner API, gunakan `EgressConfiguration` anggota `NetworkConfiguration` parameter untuk menentukan sumber daya konektor VPC untuk layanan Anda.

Gunakan tindakan App Runner API berikut untuk mengelola sumber daya Konektor VPC Anda.

- CreateVpcConnector— Membuat konektor VPC baru.
- ListVpcConnectors— Mengembalikan daftar konektor VPC yang terkait dengan Anda. Akun AWS Daftar ini mencakup deskripsi lengkap.
- DescribeVpcConnector— Mengembalikan deskripsi lengkap dari konektor VPC.

- DeleteVpcConnector— Menghapus konektor VPC. Jika Anda mencapai kuota konektor VPC untuk Anda Akun AWS, Anda mungkin perlu menghapus konektor VPC yang tidak perlu.

Untuk menerapkan aplikasi di App Runner yang memiliki akses keluar ke VPC, Anda harus terlebih dahulu membuat Konektor VPC. Anda dapat melakukan ini dengan menentukan satu atau lebih subnet dan grup keamanan untuk dikaitkan dengan aplikasi. Anda kemudian dapat mereferensikan Konektor VPC di Buat atau UpdateServicemelalui CLI, seperti yang diilustrasikan dalam contoh berikut:

```
            cat > vpc-connector.json <<EOF
{
"VpcConnectorName": "my-vpc-connector",
"Subnets": [
"subnet-a",
"subnet-b",
"subnet-c"
],
"SecurityGroups": [
"sg-1",
"sg-2"
]
}
EOF

aws apprunner create-vpc-connector \
--cli-input-json file:///vpc-connector.json

cat > service.json <<EOF

{
"ServiceName": "my-vpc-connected-service",
"SourceConfiguration": {
"ImageRepository": {
"ImageIdentifier": "<ecr-image-identifier> ",
"ImageConfiguration": {
"Port": "8000"
},
"ImageRepositoryType": "ECR"
}
},
"NetworkConfiguration": {
```

```
"EgressConfiguration": {
"EgressType": "VPC",
"VpcConnectorArn": "arn:aws:apprunner:..../my-vpc-connector"
}
}
}
EOF

aws apprunner create-service \
--cli-input-json file:///service.js
```

Observabilitas untuk layanan App Runner

AWS App Runner terintegrasi dengan beberapa AWS layanan untuk memberi Anda rangkaian alat observabilitas ekstensif untuk layanan App Runner Anda. Topik dalam Bab ini menjelaskan kemampuan ini.

Topik

- Melacak aktivitas layanan App Runner
- Melihat log App Runner dialirkan ke Log CloudWatch
- Melihat metrik layanan App Runner dilaporkan CloudWatch
- Menangani peristiwa App Runner di EventBridge
- Logging App Runner API panggilan dengan AWS CloudTrail
- Menelusuri aplikasi App Runner Anda dengan X-Ray

Melacak aktivitas layanan App Runner

AWS App Runner menggunakan daftar operasi untuk melacak aktivitas di layanan App Runner Anda. Operasi mewakili panggilan asinkron ke tindakan API, seperti membuat layanan, memperbarui konfigurasi, dan menerapkan layanan. Bagian berikut menunjukkan cara melacak aktivitas di konsol App Runner dan menggunakan API.

Lacak aktivitas layanan App Runner

Lacak aktivitas layanan App Runner menggunakan salah satu metode berikut:

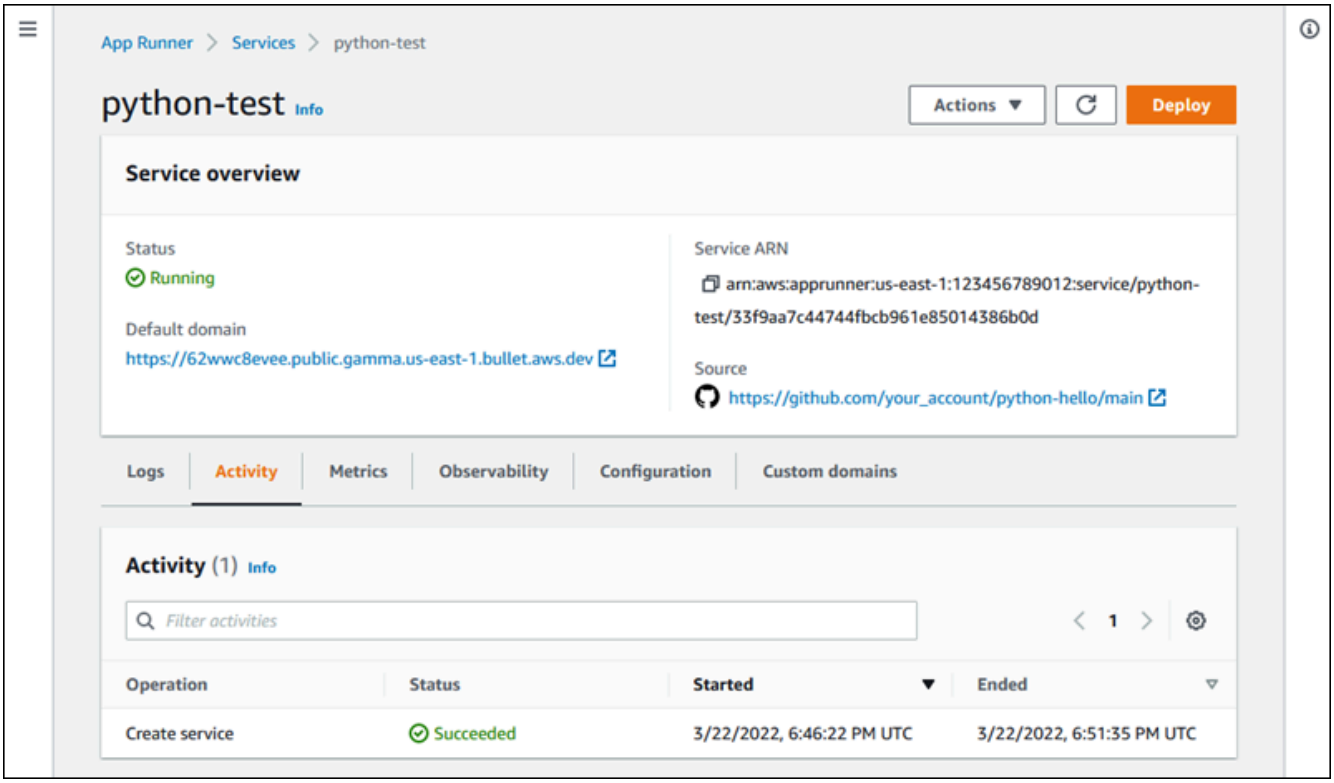
App Runner console

Konsol App Runner menampilkan aktivitas layanan App Runner Anda dan menyediakan lebih banyak cara untuk menjelajahi operasi.

Untuk melihat aktivitas layanan Anda

1. Buka konsol App Runner, dan di daftar Wilayah, pilih Wilayah AWS.
2. Di panel navigasi, pilih Layanan, lalu pilih layanan App Runner Anda.

   Konsol menampilkan dasbor layanan dengan ikhtisar Layanan.



3. Pada halaman dasbor layanan, pilih tab Aktivitas, jika belum dipilih.

   Konsol menampilkan daftar operasi.

4. Untuk menemukan operasi tertentu, lingkup daftar dengan memasukkan istilah pencarian. Anda dapat mencari nilai apa pun yang muncul di tabel.

5. Pilih operasi yang terdaftar untuk melihat atau mengunduh log terkait.

App Runner API or AWS CLI

ListOperationsTindakan, yang diberi Nama Sumber Daya Amazon (ARN) dari layanan App Runner, menampilkan daftar operasi yang terjadi pada layanan ini. Setiap item daftar berisi ID operasi dan beberapa detail pelacakan.

Melihat log App Runner dialirkan ke Log CloudWatch

Anda dapat menggunakan Amazon CloudWatch Logs untuk memantau, menyimpan, dan mengakses file log yang dihasilkan sumber daya Anda di berbagai AWS layanan. Untuk informasi selengkapnya, lihat Panduan Pengguna Amazon CloudWatch Logs.

AWS App Runner mengumpulkan output dari penerapan aplikasi Anda dan layanan aktif Anda dan mengalirkannya ke Log. CloudWatch Bagian berikut mencantumkan aliran log App Runner dan menunjukkan cara melihatnya di konsol App Runner.

Grup dan aliran log Pelari Aplikasi

CloudWatch Log menyimpan data log dalam aliran log yang selanjutnya diatur dalam grup log. Aliran log adalah urutan peristiwa log dari sumber tertentu. Grup log adalah grup log stream yang berbagi pengaturan retensi, pemantauan, dan kontrol akses yang sama.

App Runner mendefinisikan dua grup CloudWatch log Log, masing-masing dengan beberapa aliran log, untuk setiap layanan App Runner di Anda. Akun AWS

Log layanan

Grup log layanan berisi keluaran logging yang dihasilkan oleh App Runner saat mengelola layanan App Runner Anda dan bertindak di atasnya.

Nama grup log	Contoh
`/aws/apprunner/ service-name /service-id /service`	`/aws/apprunner/python-test/ac7ec8b51ff34746bcb6654e0bcb23da/service`

Dalam grup log layanan, App Runner membuat aliran log peristiwa untuk menangkap aktivitas dalam siklus hidup layanan App Runner Anda. Misalnya, ini mungkin meluncurkan aplikasi Anda atau menjedanya.

Selain itu, App Runner membuat aliran log untuk setiap operasi asinkron yang berjalan lama yang terkait dengan layanan Anda. Nama aliran log mencerminkan jenis operasi dan ID operasi tertentu.

Penyebaran adalah jenis operasi. Log penerapan berisi keluaran logging dari langkah build dan deployment yang dilakukan App Runner saat Anda membuat layanan atau menerapkan versi baru aplikasi Anda. Nama aliran log penerapan dimulai dengan`deployment/`, dan diakhiri dengan ID operasi yang melakukan penerapan. Operasi ini adalah CreateServicepanggilan untuk penerapan aplikasi awal atau StartDeploymentpanggilan untuk setiap penyebaran lebih lanjut.

Dalam log penerapan, setiap pesan log dimulai dengan awalan:

- `[AppRunner]`— Output yang dihasilkan App Runner selama penerapan.
- `[Build]`— Output dari skrip build Anda sendiri.

Nama aliran log	Contoh
`events`	N/A (nama tetap)
`operation-type` `/`*`operation-id`*	`deployment/c2c8eeedea164f45 9cf78f12a8953390`

Log aplikasi

Grup log aplikasi berisi output dari kode aplikasi Anda yang sedang berjalan.

Nama grup log	Contoh
`/aws/apprunner/` *`service-name`* `/`*`service-id`* `/application`	`/aws/apprunner/python-test/ac7ec8b51ff34746bcb6654e0bcb23da/application`

Dalam grup log aplikasi, App Runner membuat aliran log untuk setiap instance (unit penskalaan) yang menjalankan aplikasi Anda.

Nama aliran log	Contoh
`instance/` *`instance-id`*	`instance/1a80bc9134a84699b7b3432ebeebb591`

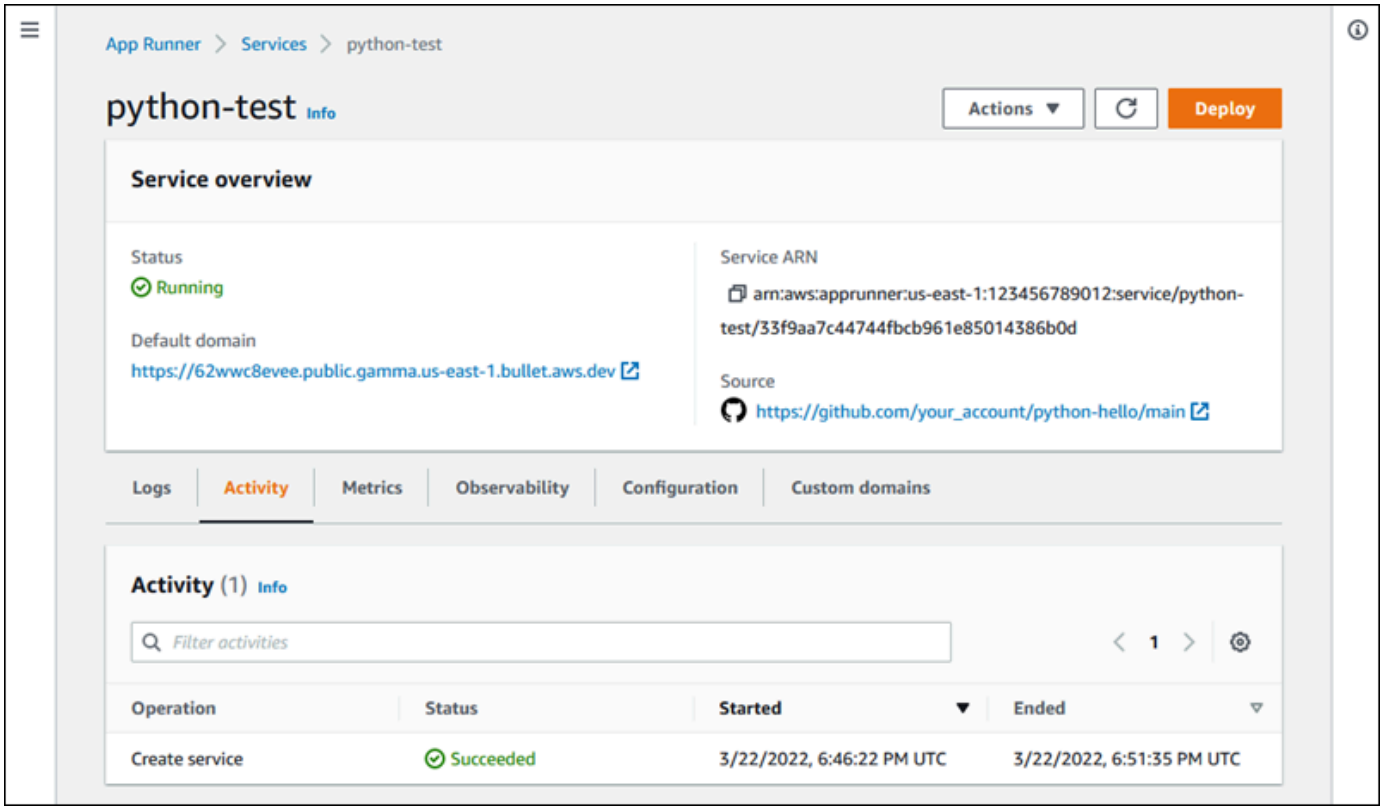
Melihat log App Runner di konsol

Konsol App Runner menampilkan ringkasan semua log untuk layanan Anda dan memungkinkan Anda untuk melihat, menjelajahi, dan mengunduhnya.

Untuk melihat log untuk layanan Anda

1. Buka konsol App Runner, dan di daftar Wilayah, pilih konsol Anda Wilayah AWS.
2. Di panel navigasi, pilih Layanan, lalu pilih layanan App Runner Anda.

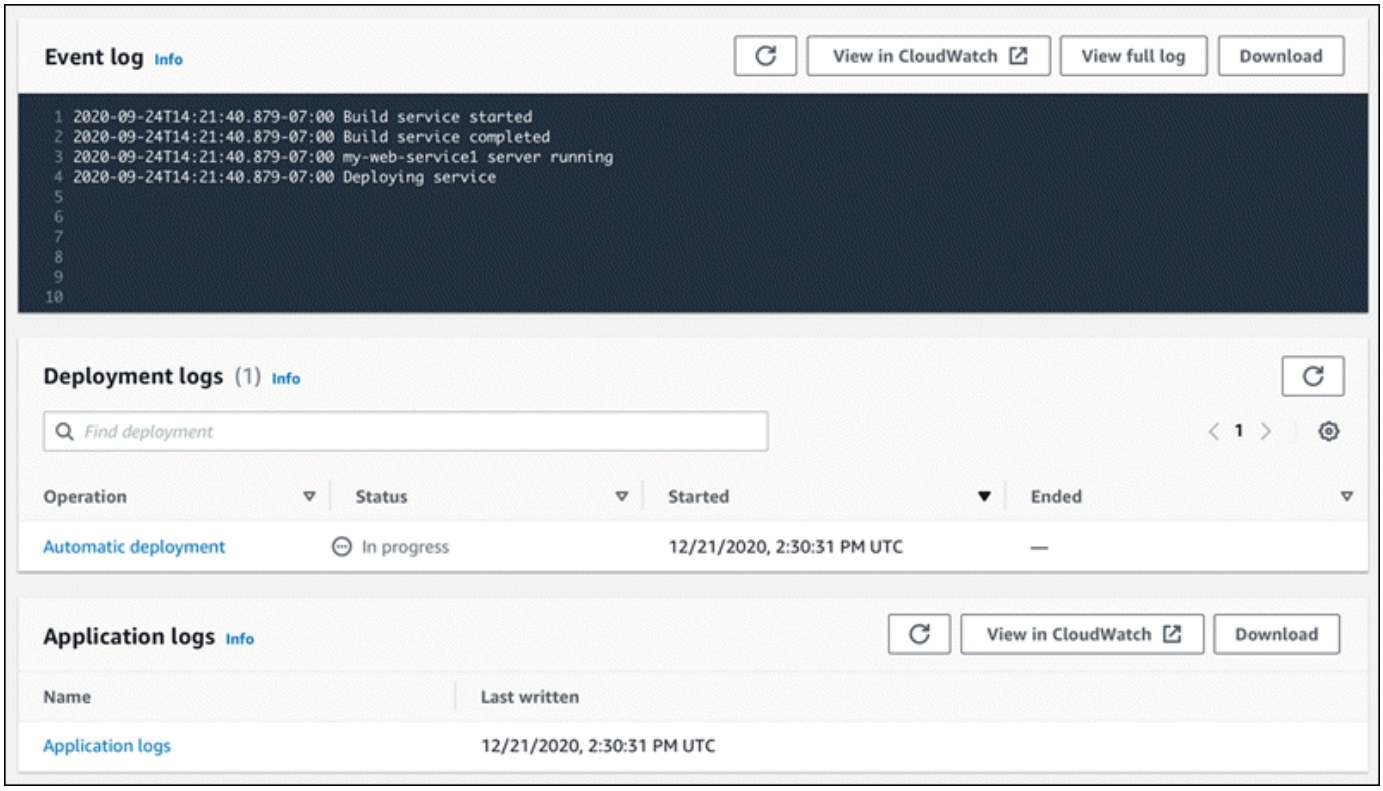
   Konsol menampilkan dasbor layanan dengan ikhtisar Layanan.



3. Pada halaman dasbor layanan, pilih tab Log.

   Konsol menampilkan beberapa jenis log di beberapa bagian:

   - Log peristiwa — Aktivitas dalam siklus hidup layanan App Runner Anda. Konsol menampilkan acara terbaru.
   - Log penerapan — Penerapan repositori sumber ke layanan App Runner Anda. Konsol menampilkan aliran log terpisah untuk setiap penerapan.
   - Log aplikasi — Output dari aplikasi web yang diterapkan ke layanan App Runner Anda. Konsol menggabungkan output dari semua instance yang berjalan ke dalam satu aliran log.



4. Untuk menemukan penerapan tertentu, cakupkan daftar log penerapan dengan memasukkan istilah penelusuran. Anda dapat mencari nilai apa pun yang muncul di tabel.
5. Untuk melihat konten log, pilih Lihat log lengkap (log peristiwa) atau nama aliran log (penerapan dan log aplikasi).
6. Pilih Unduh untuk mengunduh log. Untuk aliran log penerapan, pilih aliran log terlebih dahulu.
7. Pilih Lihat CloudWatch untuk membuka CloudWatch konsol dan gunakan kemampuan penuhnya untuk menjelajahi log layanan App Runner Anda. Untuk aliran log penerapan, pilih aliran log terlebih dahulu.

> **Note**
>
> CloudWatch Konsol ini sangat berguna jika Anda ingin melihat log aplikasi dari instance tertentu alih-alih log aplikasi gabungan.

Melihat metrik layanan App Runner dilaporkan CloudWatch

Amazon CloudWatch memantau sumber daya Amazon Web Services (AWS) Anda dan aplikasi yang Anda jalankan AWS secara real time. Anda dapat menggunakan CloudWatch untuk mengumpulkan dan melacak metrik, yang merupakan variabel yang dapat Anda ukur untuk sumber daya dan aplikasi Anda. Anda juga dapat menggunakannya untuk membuat alarm yang menonton metrik. Ketika ambang batas tertentu tercapai, CloudWatch mengirim pemberitahuan, atau secara otomatis membuat perubahan pada sumber daya yang dipantau. Untuk informasi selengkapnya, lihat Panduan CloudWatch Pengguna Amazon.

AWS App Runner mengumpulkan berbagai metrik yang memberi Anda visibilitas yang lebih besar terhadap penggunaan, kinerja, dan ketersediaan layanan Pelari Aplikasi Anda. Beberapa metrik melacak instance individual yang menjalankan layanan web Anda, sedangkan yang lain berada di tingkat layanan keseluruhan. Bagian berikut mencantumkan metrik App Runner dan menunjukkan cara melihatnya di konsol App Runner.

Metrik Pelari Aplikasi

App Runner mengumpulkan metrik berikut yang berkaitan dengan layanan Anda dan menerbitkannya di namespace. CloudWatch `AWS/AppRunner`

> **Note**
>
> Sebelum 23 Agustus 2023, metrik pemanfaatan CPU dan pemanfaatan Memori didasarkan pada unit vCPU dan megabita memori yang digunakan, bukan persen pemanfaatan, seperti yang dihitung hari ini. Jika aplikasi Anda berjalan di App Runner sebelum tanggal ini, dan Anda memilih untuk kembali melihat metrik untuk tanggal ini di App Runner atau CloudWatch konsol, Anda akan melihat tampilan metrik di kedua unit dan juga akan melihat beberapa penyimpangan sebagai hasilnya.

> **Important**
>
> Anda harus memperbarui CloudWatch alarm apa pun yang didasarkan pada pemanfaatan CPU dan nilai metrik pemanfaatan Memori sebelum 23 Agustus 2023. Perbarui alarm untuk memicu berdasarkan persentase pemanfaatan daripada vCPU atau megabyte. Untuk informasi selengkapnya, lihat Panduan CloudWatch Pengguna Amazon.

Metrik tingkat instans dikumpulkan untuk setiap instance (unit penskalaan) secara individual.

Apa yang diukur?	Metrik	Deskripsi
CPU utilization	`CPUUtilization`	Persentase penggunaan CPU rata-rata selama periode satu menit dari total penggunaan CPU yang dicadangkan oleh konfigurasi layanan.
Memory utilization	`MemoryUtilization`	Persentase penggunaan memori rata-rata selama periode satu menit dari total memori yang dicadangkan oleh konfigurasi layanan.

Metrik tingkat layanan dikumpulkan untuk seluruh layanan.

Apa yang diukur?	Metrik	Deskripsi
CPU utilization	`CPUUtilization`	Persentase penggunaan CPU agregat di semua instance selama periode satu menit dari total penggunaan CPU yang dicadangkan oleh konfigurasi layanan.
Memory utilization	`MemoryUtilization`	Persentase penggunaan memori agregat di semua instance selama periode satu menit dari total memori yang dicadangkan oleh konfigurasi layanan.
Concurrency	`Concurrency`	Perkiraan jumlah permintaan bersamaan yang ditangani oleh layanan.
HTTP request count	`Requests`	Jumlah permintaan HTTP yang diterima layanan.
HTTP status counts	`2xxStatusResponses`	Jumlah permintaan HTTP yang mengembalikan setiap status respons, dikelompokkan berdasarkan kategori (2XX, 4XX, 5XX).

Apa yang diukur?	Metrik	Deskripsi

	`4xxStatusResponses`
	
	`5xxStatusResponses`	

HTTP request latency	`RequestLatency`	Waktu, dalam milidetik, dibutuhkan layanan web Anda untuk memproses permintaan HTTP.
Instance counts	`ActiveInstances`	Jumlah instance yang memproses permintaan HTTP untuk layanan Anda. **Note** Jika `ActiveInstances` metrik menampilkan nol, itu berarti tidak ada permintaan untuk layanan. Ini tidak menunjukkan bahwa jumlah instance untuk layanan Anda adalah nol.

Melihat metrik App Runner di konsol

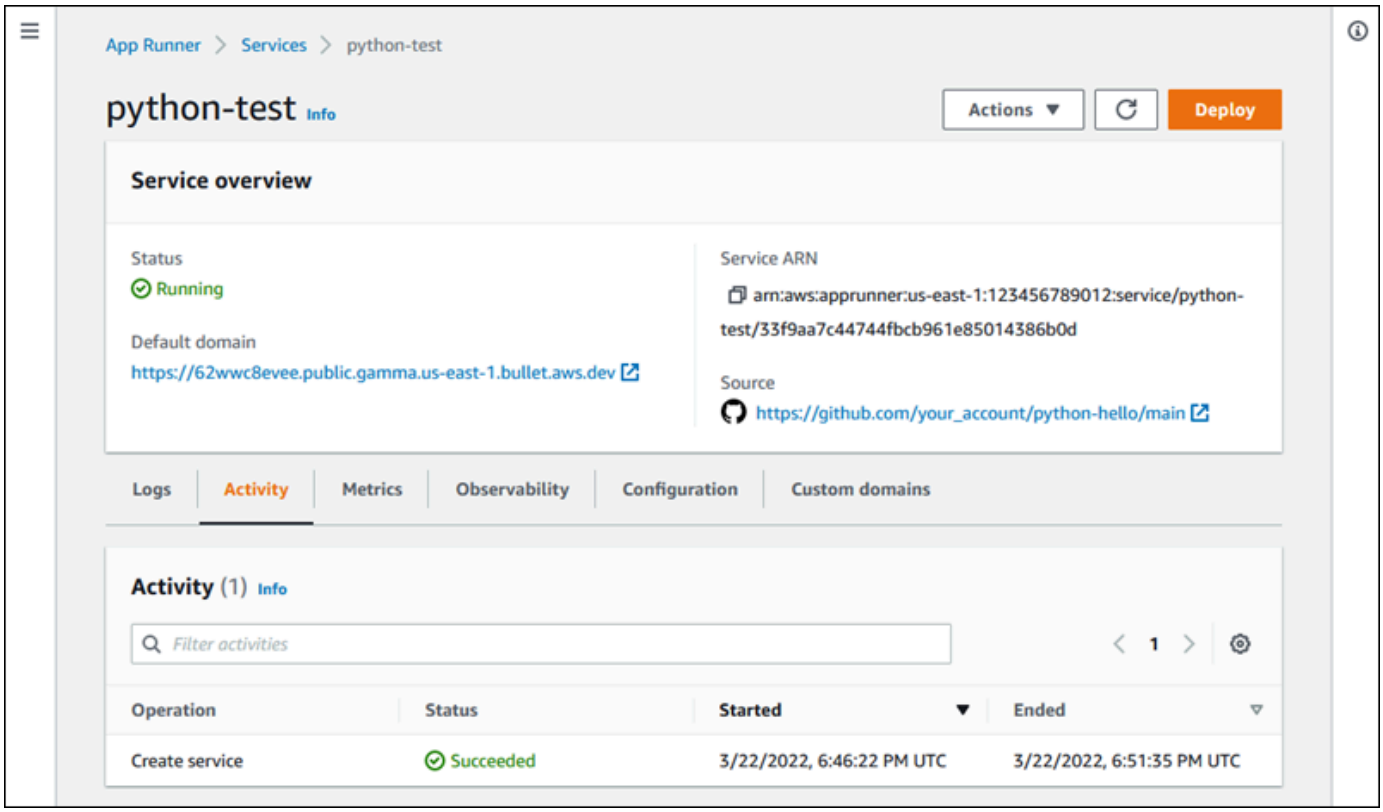
Konsol App Runner secara grafis menampilkan metrik yang dikumpulkan App Runner untuk layanan Anda dan menyediakan lebih banyak cara untuk menjelajahinya.

> **Note**
> Pada saat ini, konsol hanya menampilkan metrik layanan. Untuk melihat metrik instance, gunakan CloudWatch konsol.

Untuk melihat log untuk layanan Anda

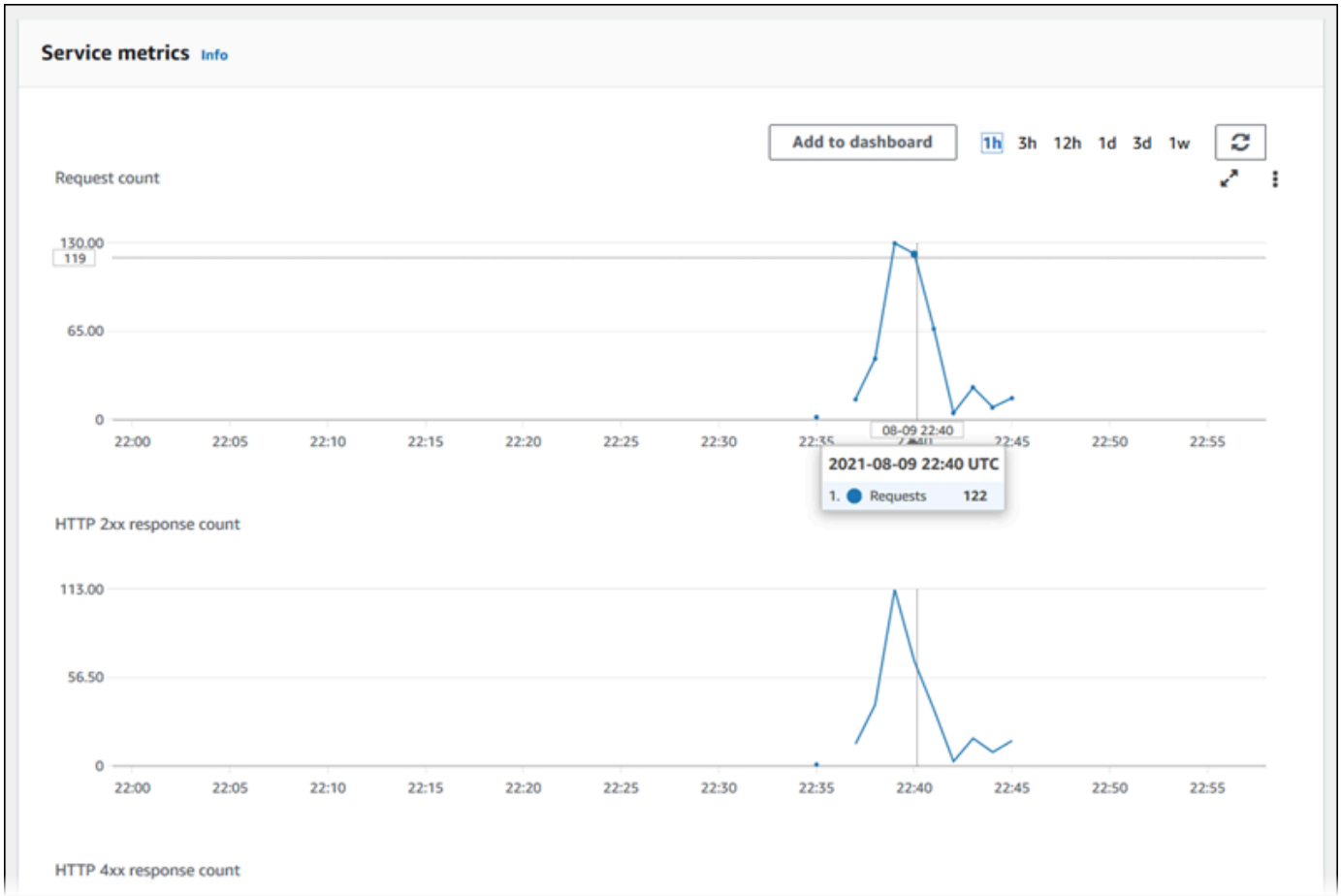
1. Buka konsol App Runner, dan di daftar Wilayah, pilih konsol Anda Wilayah AWS.
2. Di panel navigasi, pilih Layanan, lalu pilih layanan App Runner Anda.

Konsol menampilkan dasbor layanan dengan ikhtisar Layanan.



3. Pada halaman dasbor layanan, pilih tab Metrik.

   Konsol menampilkan satu set grafik metrik.



4. Pilih durasi (misalnya, 12 jam) untuk cakupan grafik metrik ke periode terakhir dari durasi tersebut.
5. Pilih Tambahkan ke dasbor di bagian atas salah satu bagian grafik, atau gunakan menu pada grafik apa pun, untuk menambahkan metrik yang relevan ke dasbor di CloudWatch konsol untuk penyelidikan lebih lanjut.

Menangani peristiwa App Runner di EventBridge

Menggunakan Amazon EventBridge, Anda dapat mengatur aturan berbasis peristiwa yang memantau aliran data real-time dari AWS App Runner layanan Anda untuk pola tertentu. Ketika pola untuk aturan dicocokkan, EventBridge memulai tindakan dalam target seperti, Amazon ECS AWS Batch, dan AWS Lambda Amazon SNS. Misalnya, Anda dapat menetapkan aturan untuk mengirimkan pemberitahuan email dengan memberi sinyal topik Amazon SNS setiap kali penerapan ke layanan Anda gagal. Atau, Anda dapat mengatur fungsi Lambda untuk memberi tahu saluran Slack setiap kali pembaruan layanan gagal. Untuk informasi selengkapnya EventBridge, lihat Panduan EventBridge Pengguna Amazon.

App Runner mengirimkan jenis acara berikut ke EventBridge

- Perubahan status layanan — Perubahan status layanan App Runner. Misalnya, status layanan diubah menjadi`DELETE_FAILED`.
- Perubahan status operasi layanan — Perubahan status operasi asinkron yang panjang pada layanan App Runner. Misalnya, layanan mulai dibuat, pembaruan layanan berhasil diselesaikan, atau penyebaran layanan yang dilengkapi dengan kesalahan.

Membuat EventBridge aturan untuk bertindak pada acara App Runner

EventBridge Peristiwa adalah objek yang mendefinisikan beberapa EventBridge bidang standar, seperti AWS layanan sumber dan jenis detail (peristiwa), dan kumpulan bidang khusus peristiwa dengan detail acara. Untuk membuat EventBridge aturan, Anda menggunakan EventBridge konsol untuk menentukan pola acara (acara mana yang harus dilacak) dan menentukan tindakan target (apa yang harus dilakukan pada pertandingan). Pola acara mirip dengan peristiwa yang cocok. Anda menentukan subset bidang yang cocok, dan untuk setiap bidang, Anda menentukan daftar nilai yang mungkin. Topik ini memberikan contoh peristiwa App Runner dan pola acara.

Untuk informasi selengkapnya tentang membuat EventBridge aturan, lihat Membuat aturan untuk AWS layanan di Panduan EventBridge Pengguna Amazon.

> **Note**
>
> Beberapa layanan mendukung pola yang telah ditentukan sebelumnya di EventBridge. Ini menyederhanakan bagaimana pola acara dibuat. Anda memilih nilai bidang pada formulir, dan EventBridge menghasilkan pola untuk Anda. Saat ini, App Runner tidak mendukung pola yang telah ditentukan sebelumnya. Anda harus memasukkan pola sebagai objek JSON. Anda dapat menggunakan contoh dalam topik ini sebagai titik awal.

Contoh acara App Runner

Ini adalah beberapa contoh acara yang dikirimkan oleh App Runner. EventBridge

- Peristiwa perubahan status layanan. Secara khusus, layanan yang berubah dari `RUNNING` status `OPERATION_IN_PROGRESS` ke status.

```
{
```

```
  "version": "0",
  "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
  "detail-type": "AppRunner Service Status Change",
  "source": "aws.apprunner",
  "account": "111122223333",
  "time": "2021-04-29T11:54:23Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:apprunner:us-east-2:123456789012:service/my-
app/8fe1e10304f84fd2b0df550fe98a71fa"
  ],
  "detail": {
    "previousServiceStatus": "OPERATION_IN_PROGRESS",
    "currentServiceStatus": "RUNNING",
    "serviceName": "my-app",
    "serviceId": "8fe1e10304f84fd2b0df550fe98a71fa",
    "message": "Service status is set to RUNNING.",
    "severity": "INFO"
  }
}
```

- Peristiwa perubahan status operasi. Secara khusus, `UpdateService` operasi yang berhasil diselesaikan.

```
{
  "version": "0",
  "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
  "detail-type": "AppRunner Service Operation Status Change",
  "source": "aws.apprunner",
  "account": "111122223333",
  "time": "2021-04-29T18:43:48Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:apprunner:us-east-2:123456789012:service/my-
app/8fe1e10304f84fd2b0df550fe98a71fa"
  ],
  "detail": {
    "operationStatus": "UpdateServiceCompletedSuccessfully",
    "serviceName": "my-app",
    "serviceId": "8fe1e10304f84fd2b0df550fe98a71fa",
    "message": "Service update completed successfully. New application and
 configuration is deployed.",
    "severity": "INFO"
```

```
  }
}
```

Contoh pola acara App Runner

Contoh berikut menunjukkan pola peristiwa yang dapat Anda gunakan dalam EventBridge aturan untuk mencocokkan satu atau beberapa peristiwa App Runner. Pola peristiwa mirip dengan suatu peristiwa. Sertakan hanya bidang yang ingin Anda cocokkan, dan berikan daftar alih-alih skalar untuk masing-masing bidang.

- Cocokkan semua peristiwa perubahan status layanan untuk layanan akun tertentu, di mana layanan tidak lagi dalam `RUNNING` status.

```
{
  "detail-type": [ "AppRunner Service Status Change" ],
  "source": [ "aws.apprunner" ],
  "account": [ "111122223333" ],
  "detail": {
    "previousServiceStatus": [ "RUNNING" ]
  }
}
```

- Cocokkan semua peristiwa perubahan status operasi untuk layanan akun tertentu, di mana operasi gagal.

```
{
  "detail-type": [ "AppRunner Service Operation Status Change" ],
  "source": [ "aws.apprunner" ],
  "account": [ "111122223333" ],
  "detail": {
    "operationStatus": [
      "CreateServiceFailed",
      "DeleteServiceFailed",
      "UpdateServiceFailed",
      "DeploymentFailed",
      "PauseServiceFailed",
      "ResumeServiceFailed"
    ]
  }
}
```

Referensi acara App Runner

Perubahan status layanan

Peristiwa perubahan status layanan telah `detail-type` disetel ke`AppRunner Service Status Change`. Ini memiliki bidang dan nilai detail berikut:

```
"serviceId": "your service ID",
"serviceName": "your service name",
"message": "Service status is set to CurrentStatus.",
"previousServiceStatus": "any valid service status",
"currentServiceStatus": "any valid service status",
"severity": "varies"
```

Perubahan status operasi

Peristiwa perubahan status operasi telah `detail-type` disetel ke`AppRunner Service Operation Status Change`. Ini memiliki bidang dan nilai detail berikut:

```
"operationStatus": "see following table",
"serviceName": "your service name",
"serviceId": "your service ID",
"message": "see following table",
"severity": "varies"
```

Tabel berikut mencantumkan semua kode status yang mungkin dan pesan terkait.

Status	Pesan
`CreateServiceStarted`	Pembuatan layanan dimulai.
`CreateServiceCompletedSuccessfully`	Pembuatan layanan selesai dengan sukses.
`CreateServiceFailed`	Pembuatan layanan gagal. Untuk detailnya, lihat log layanan.
`DeleteServiceStarted`	Penghapusan layanan dimulai.
`DeleteServiceCompletedSuccessfully`	Penghapusan layanan berhasil diselesaikan.

Status	Pesan
`DeleteServiceFailed`	Penghapusan layanan gagal.
`UpdateServiceStarted`	
`UpdateServiceCompletedSuccessfully`	Pembaruan layanan berhasil diselesaikan. Aplikasi dan konfigurasi baru diterapkan.
	Pembaruan layanan berhasil diselesaikan. Konfigurasi baru diterapkan.
`UpdateServiceFailed`	Pembaruan layanan gagal. Untuk detailnya, lihat log layanan.
`DeploymentStarted`	Penerapan dimulai.
`DeploymentCompletedSuccessfully`	Penerapan selesai dengan sukses.
`DeploymentFailed`	Deployment gagal. Untuk detailnya, lihat log layanan.
`PauseServiceStarted`	Jeda layanan dimulai.
`PauseServiceCompletedSuccessfully`	Jeda layanan selesai dengan sukses.
`PauseServiceFailed`	Jeda layanan gagal.
`ResumeServiceStarted`	Resume layanan dimulai.
`ResumeServiceCompletedSuccessfully`	Resume layanan selesai dengan sukses.
`ResumeServiceFailed`	Resume layanan gagal.

Logging App Runner API panggilan dengan AWS CloudTrail

App Runner terintegrasi dengan AWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan di App Runner. CloudTrailmenangkap semua panggilan API untuk App Runner sebagai peristiwa. Panggilan yang diambil termasuk panggilan

dari konsol App Runner dan panggilan kode ke operasi App Runner API. Jika Anda membuat jejak, Anda dapat mengaktifkan pengiriman CloudTrail acara secara terus menerus ke bucket Amazon S3, termasuk acara untuk App Runner. Jika Anda tidak mengonfigurasi jejak, Anda masih dapat melihat peristiwa terbaru di CloudTrail konsol dalam Riwayat acara. Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat untuk App Runner, alamat IP dari tempat permintaan dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan.

Untuk mempelajari selengkapnya CloudTrail, lihat Panduan AWS CloudTrail Pengguna.

Informasi Pelari Aplikasi di CloudTrail

CloudTrail diaktifkan pada Akun AWS saat Anda membuat akun. Saat aktivitas terjadi di App Runner, aktivitas tersebut direkam dalam suatu CloudTrail peristiwa bersama dengan peristiwa AWS layanan lainnya dalam riwayat Acara. Anda dapat melihat, mencari, dan mengunduh acara terbaru di situs Anda Akun AWS. Untuk informasi selengkapnya, lihat Melihat Acara dengan Riwayat CloudTrail Acara.

Untuk catatan acara yang sedang berlangsung di Anda Akun AWS, termasuk acara untuk App Runner, buat jejak. Jejak memungkinkan CloudTrail untuk mengirimkan file log ke bucket Amazon S3. Secara default, saat Anda membuat jejak di konsol, jejak tersebut berlaku untuk semua Wilayah AWS. Jejak mencatat peristiwa dari semua Wilayah di AWS partisi dan mengirimkan file log ke bucket Amazon S3 yang Anda tentukan. Selain itu, Anda dapat mengonfigurasi AWS layanan lain untuk menganalisis lebih lanjut dan menindaklanjuti data peristiwa yang dikumpulkan dalam CloudTrail log. Untuk informasi selengkapnya, lihat berikut:

- Gambaran Umum untuk Membuat Jejak
- CloudTrail Layanan dan Integrasi yang Didukung
- Mengkonfigurasi Notifikasi Amazon SNS untuk CloudTrail
- Menerima File CloudTrail Log dari Beberapa Wilayah dan Menerima File CloudTrail Log dari Beberapa Akun

Semua tindakan App Runner dicatat oleh CloudTrail dan didokumentasikan dalam Referensi AWS App Runner API. Misalnya, panggilan ke`CreateService`,`DeleteConnection`, dan `StartDeployment` tindakan menghasilkan entri dalam file CloudTrail log.

Setiap entri peristiwa atau log berisi informasi tentang siapa yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan hal berikut ini:

- Apakah permintaan tersebut dibuat dengan kredensial root atau pengguna IAM.
- Apakah permintaan tersebut dibuat dengan kredensial keamanan sementara untuk satu peran atau pengguna terfederasi.
- Apakah permintaan itu dibuat oleh AWS layanan lain.

Untuk informasi lain, lihat Elemen userIdentity CloudTrail .

Memahami entri file log App Runner

Trail adalah konfigurasi yang memungkinkan pengiriman peristiwa sebagai file log ke bucket Amazon S3 yang Anda tentukan. CloudTrail file log berisi satu atau lebih entri log. Peristiwa mewakili permintaan tunggal dari sumber manapun dan mencakup informasi tentang tindakan yang diminta, tanggal dan waktu tindakan, dan parameter permintaan. CloudTrail file log bukanlah jejak tumpukan yang diurutkan dari panggilan API publik, jadi file tersebut tidak muncul dalam urutan tertentu.

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan `CreateService` tindakan.

> **Note**
>
> Untuk alasan keamanan, beberapa nilai properti disunting di log dan diganti dengan teks`HIDDEN_DUE_TO_SECURITY_REASONS`. Ini mencegah paparan informasi rahasia yang tidak diinginkan. Namun, Anda masih dapat melihat bahwa properti ini diteruskan dalam permintaan atau dikembalikan dalam respons.

Contoh entri CloudTrail log untuk tindakan **`CreateService`** App Runner

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:iam::123456789012:user/aws-user",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "aws-user"
  },
  "eventTime": "2020-10-02T23:25:33Z",
  "eventSource": "apprunner.amazonaws.com",
  "eventName": "CreateService",
```

```
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_6) AppleWebKit/537.36
 (KHTML, like Gecko) Chrome/77.0.3865.75 Safari/537.36",
  "requestParameters": {
    "serviceName": "python-test",
    "sourceConfiguration": {
      "codeRepository": {
        "repositoryUrl": "https://github.com/github-user/python-hello",
        "sourceCodeVersion": {
          "type": "BRANCH",
          "value": "main"
        },
        "codeConfiguration": {
          "configurationSource": "API",
          "codeConfigurationValues": {
            "runtime": "python3",
            "buildCommand": "HIDDEN_DUE_TO_SECURITY_REASONS",
            "startCommand": "HIDDEN_DUE_TO_SECURITY_REASONS",
            "port": "8080",
            "runtimeEnvironmentVariables": "HIDDEN_DUE_TO_SECURITY_REASONS"
          }
        }
      },
      "autoDeploymentsEnabled": true,
      "authenticationConfiguration": {
        "connectionArn": "arn:aws:apprunner:us-east-2:123456789012:connection/your-
connection/e7656250f67242d7819feade6800f59e"
      }
    },
    "healthCheckConfiguration": {
      "protocol": "HTTP"
    },
    "instanceConfiguration": {
      "cpu": "256",
      "memory": "1024"
    }
  },
  "responseElements": {
    "service": {
        "serviceName": "python-test",
        "serviceId": "dfa2b7cc7bcb4b6fa6c1f0f4efff988a",
        "serviceArn": "arn:aws:apprunner:us-east-2:123456789012:service/python-test/
dfa2b7cc7bcb4b6fa6c1f0f4efff988a",
```

```
        "serviceUrl": "generated domain",
        "createdAt": "2020-10-02T23:25:32.650Z",
        "updatedAt": "2020-10-02T23:25:32.650Z",
        "status": "OPERATION_IN_PROGRESS",
        "sourceConfiguration": {
            "codeRepository": {
                "repositoryUrl": "https://github.com/github-user/python-hello",
                "sourceCodeVersion": {
                    "type": "Branch",
                    "value": "main"
                },
                "sourceDirectory": "/",
                "codeConfiguration": {
                    "codeConfigurationValues": {
                        "configurationSource": "API",
                        "runtime": "python3",
                        "buildCommand": "HIDDEN_DUE_TO_SECURITY_REASONS",
                        "startCommand": "HIDDEN_DUE_TO_SECURITY_REASONS",
                        "port": "8080",
                        "runtimeEnvironmentVariables": "HIDDEN_DUE_TO_SECURITY_REASONS"
                    }
                }
            },
            "autoDeploymentsEnabled": true,
            "authenticationConfiguration": {
                "connectionArn": "arn:aws:apprunner:us-east-2:123456789012:connection/
your-connection/e7656250f67242d7819feade6800f59e"
            }
        },
        "healthCheckConfiguration": {
            "protocol": "HTTP",
            "path": "/",
            "interval": 5,
            "timeout": 2,
            "healthyThreshold": 3,
            "unhealthyThreshold": 5
        },
        "instanceConfiguration": {
            "cpu": "256",
            "memory": "1024"
        },
        "autoScalingConfigurationSummary": {
```

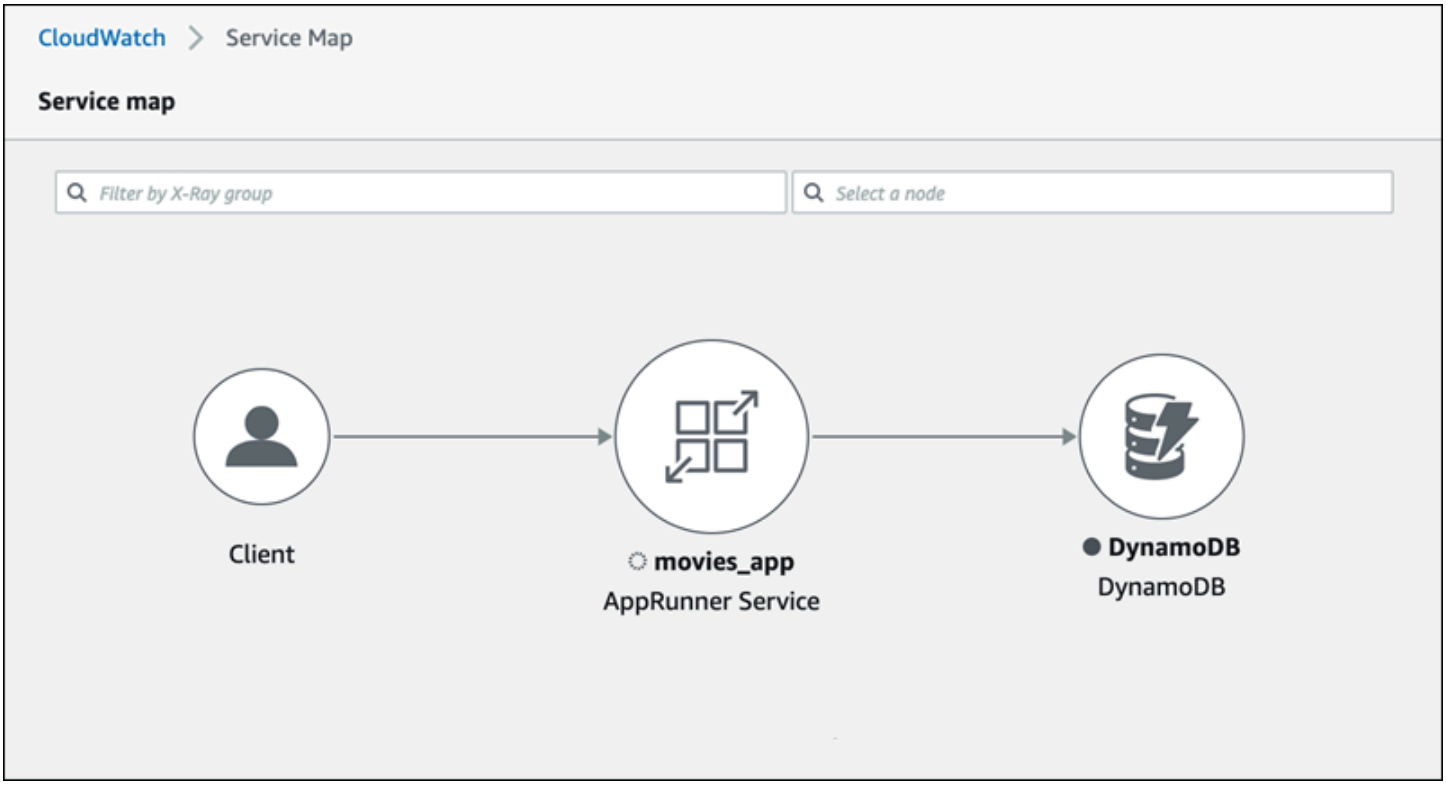
```
            "autoScalingConfigurationArn": "arn:aws:apprunner:us-
east-2:123456789012:autoscalingconfiguration/
DefaultConfiguration/1/00000000000000000000000000000001",
            "autoScalingConfigurationName": "DefaultConfiguration",
            "autoScalingConfigurationRevision": 1
        }
    }
},
  "requestID": "1a60af60-ecf5-4280-aa8f-64538319ba0a",
  "eventID": "e1a3f623-4d24-4390-a70b-bf08a0e24669",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```

Menelusuri aplikasi App Runner Anda dengan X-Ray

AWS X-Ray adalah layanan yang mengumpulkan data tentang permintaan yang disajikan aplikasi Anda, dan menyediakan alat yang dapat Anda gunakan untuk melihat, memfilter, dan mendapatkan wawasan tentang data tersebut guna mengidentifikasi masalah dan peluang pengoptimalan. Untuk setiap permintaan yang dilacak ke aplikasi Anda, Anda dapat melihat informasi terperinci tidak hanya tentang permintaan dan respons, tetapi juga tentang panggilan yang dilakukan aplikasi Anda ke AWS sumber daya hilir, layanan mikro, database, dan web HTTP. APIs

X-Ray menggunakan data jejak dari AWS sumber daya yang memberi daya pada aplikasi cloud Anda untuk menghasilkan grafik layanan terperinci. Grafik layanan menunjukkan klien, layanan front-end Anda, dan layanan backend yang dipanggil layanan front-end Anda untuk memproses permintaan dan menyimpan data. Gunakan grafik layanan untuk mengidentifikasi hambatan, lonjakan latensi, dan masalah lain yang perlu dipecahkan untuk meningkatkan performa aplikasi Anda.

Untuk informasi selengkapnya tentang X-Ray, lihat Panduan Developer AWS X-Ray.



Instrumen aplikasi Anda untuk melacak

Instrumentasikan aplikasi layanan App Runner Anda untuk melacak menggunakan OpenTelemetry, spesifikasi telemetri portabel. Pada saat ini, App Runner mendukung AWS Distro for OpenTelemetry (ADOT), sebuah OpenTelemetry implementasi yang mengumpulkan dan menyajikan informasi telemetri menggunakan layanan. AWS X-Ray mengimplementasikan komponen penelusuran.

Bergantung pada SDK ADOT spesifik yang Anda gunakan dalam aplikasi Anda, ADOT mendukung hingga dua pendekatan instrumentasi: otomatis dan manual. Untuk informasi selengkapnya tentang instrumentasi dengan SDK Anda, lihat dokumentasi ADOT, dan pilih SDK Anda di panel navigasi.

Pengaturan runtime

Berikut ini adalah petunjuk penyiapan runtime umum untuk menginstruksikan aplikasi layanan App Runner Anda untuk dilacak.

Untuk mengatur penelusuran untuk runtime Anda

1. Ikuti petunjuk yang diberikan untuk runtime Anda di AWS Distro for OpenTelemetry (ADOT), untuk instrumen aplikasi Anda.

2. Instal `OTEL` dependensi yang diperlukan di `build` bagian `apprunner.yaml` file jika Anda menggunakan repositori kode sumber atau di Dockerfile jika Anda menggunakan gambar kontainer.
3. Siapkan variabel lingkungan Anda dalam `apprunner.yaml` file jika Anda menggunakan repositori kode sumber atau di Dockerfile jika Anda menggunakan gambar kontainer.

   Example Variabel-variabel lingkungan

> **Note**
>
> Contoh berikut mencantumkan variabel lingkungan penting untuk ditambahkan ke `apprunner.yaml` file. Tambahkan variabel lingkungan ini ke Dockerfile Anda jika Anda menggunakan gambar kontainer. Namun, setiap runtime dapat memiliki keistimewaannya sendiri dan Anda mungkin perlu menambahkan lebih banyak variabel lingkungan ke daftar berikut. Untuk informasi selengkapnya tentang instruksi dan contoh spesifik runtime Anda tentang cara mengatur aplikasi untuk runtime Anda, lihat AWS Distro untuk OpenTelemetry dan buka runtime Anda, di bawah Memulai.

```
env:
    - name: OTEL_PROPAGATORS
      value: xray
    - name: OTEL_METRICS_EXPORTER
      value: none
    - name: OTEL_EXPORTER_OTLP_ENDPOINT
      value: http://localhost:4317
    - name: OTEL_RESOURCE_ATTRIBUTES
      value: 'service.name=example_app'
```

> **Note**
>
> `OTEL_METRICS_EXPORTER=none`adalah variabel lingkungan penting untuk App Runner karena kolektor App Runner Otel tidak menerima pencatatan metrik. Itu hanya menerima penelusuran metrik.

Contoh pengaturan runtime

Contoh berikut menunjukkan auto-instrumentasi aplikasi Anda dengan ADOT Python SDK. SDK secara otomatis menghasilkan bentang dengan data telemetri yang menjelaskan nilai yang digunakan oleh framework Python dalam aplikasi Anda tanpa menambahkan satu baris kode Python. Anda perlu menambahkan atau memodifikasi hanya beberapa baris dalam dua file sumber.

Pertama, tambahkan beberapa dependensi, seperti yang ditunjukkan pada contoh berikut.

Example requirements.txt

```
opentelemetry-distro[otlp]>=0.24b0
opentelemetry-sdk-extension-aws~=2.0
opentelemetry-propagator-aws-xray~=1.0
```

Kemudian, instrumen aplikasi Anda. Cara melakukannya tergantung pada sumber layanan Anda — gambar sumber atau kode sumber.

Source image

Ketika sumber layanan Anda adalah gambar, Anda dapat langsung instrumen Dockerfile yang mengontrol pembuatan image kontainer Anda dan menjalankan aplikasi dalam gambar. Contoh berikut menunjukkan Dockerfile instrumentasi untuk aplikasi Python. Penambahan instrumentasi ditekankan dalam huruf tebal.

Example Dockerfile

```
FROM public.ecr.aws/amazonlinux/amazonlinux:latest
RUN yum install python3.7 -y && curl -O https://bootstrap.pypa.io/get-pip.py &&
 python3 get-pip.py && yum update -y
COPY . /app
WORKDIR /app
RUN pip3 install -r requirements.txt
RUN opentelemetry-bootstrap --action=install
ENV OTEL_PYTHON_DISABLED_INSTRUMENTATIONS=urllib3
ENV OTEL_METRICS_EXPORTER=none
ENV OTEL_RESOURCE_ATTRIBUTES='service.name=example_app'
CMD OTEL_PROPAGATORS=xray OTEL_PYTHON_ID_GENERATOR=xray opentelemetry-instrument
 python3 app.py
EXPOSE 8080
```

Source code repository

Ketika sumber layanan Anda adalah repositori yang berisi sumber aplikasi Anda, Anda secara tidak langsung instrumen gambar Anda menggunakan pengaturan file konfigurasi App Runner. Pengaturan ini mengontrol Dockerfile yang dihasilkan dan digunakan App Runner untuk membangun gambar untuk aplikasi Anda. Contoh berikut menunjukkan file konfigurasi App Runner yang diinstrumentasi untuk aplikasi Python. Penambahan instrumentasi ditekankan dalam huruf tebal.

Example apprunner.yaml

```
version: 1.0
runtime: python3
build:
  commands:
    build:
      - pip install -r requirements.txt
      - opentelemetry-bootstrap --action=install
run:
  command: opentelemetry-instrument python app.py
  network:
    port: 8080
  env:
    - name: OTEL_PROPAGATORS
      value: xray
    - name: OTEL_METRICS_EXPORTER
      value: none
    - name: OTEL_PYTHON_ID_GENERATOR
      value: xray
    - name: OTEL_PYTHON_DISABLED_INSTRUMENTATIONS
      value: urllib3
    - name: OTEL_RESOURCE_ATTRIBUTES
      value: 'service.name=example_app'
```

Menambahkan izin X-Ray ke peran instance layanan App Runner

Untuk menggunakan penelusuran X-Ray dengan layanan App Runner, Anda harus memberikan izin pada instance layanan untuk berinteraksi dengan layanan X-Ray. Anda melakukannya dengan mengaitkan peran instance dengan layanan Anda dan menambahkan kebijakan terkelola dengan izin X-Ray. Untuk informasi selengkapnya tentang peran instans App Runner, lihatthe section called

“Peran contoh”. Tambahkan kebijakan `AWSXRayDaemonWriteAccess` terkelola ke peran instans Anda dan tetapkan ke layanan Anda selama pembuatan.

Aktifkan penelusuran X-Ray untuk layanan App Runner

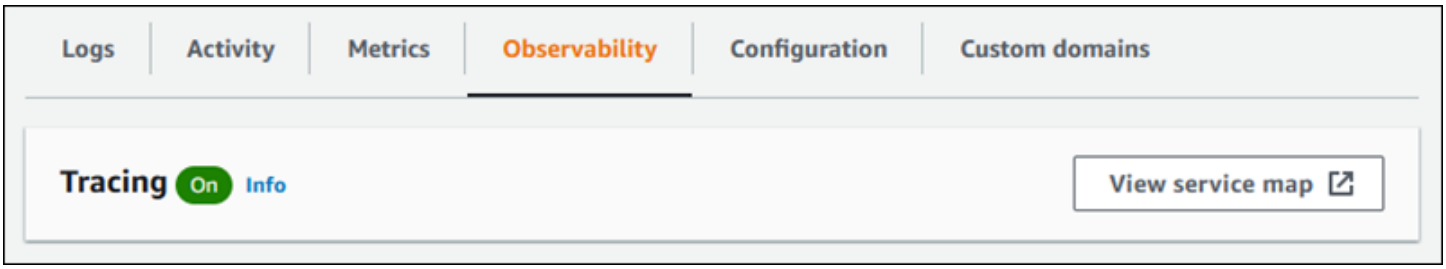
Saat Anda membuat layanan, App Runner menonaktifkan penelusuran secara default. Anda dapat mengaktifkan penelusuran X-Ray untuk layanan Anda sebagai bagian dari konfigurasi observabilitas. Untuk informasi selengkapnya, lihat the section called “Kelola observabilitas”.

Jika Anda menggunakan App Runner API atau AWS CLI, TraceConfigurationobjek dalam objek ObservabilityConfigurationresource berisi pengaturan penelusuran. Agar penelusuran tetap dinonaktifkan, jangan tentukan `TraceConfiguration` objek.

Dalam kasus konsol dan API, pastikan untuk mengaitkan peran instance yang dibahas di bagian sebelumnya dengan layanan App Runner Anda.

Melihat data penelusuran X-Ray untuk layanan App Runner

Pada tab Observability pada halaman dasbor layanan di konsol App Runner, pilih Lihat peta layanan untuk menavigasi ke konsol Amazon CloudWatch .



Gunakan CloudWatch konsol Amazon untuk melihat peta layanan dan melacak permintaan yang dilayani aplikasi Anda. Peta layanan menampilkan informasi seperti latensi permintaan dan interaksi dengan aplikasi dan AWS layanan lain. Anotasi khusus yang Anda tambahkan ke kode Anda memungkinkan Anda untuk dengan mudah mencari jejak. Untuk informasi selengkapnya, lihat Menggunakan ServiceLens untuk memantau kesehatan aplikasi Anda di Panduan CloudWatch Pengguna Amazon.

Mengaitkan ACL AWS WAF web dengan layanan Anda

AWS WAF adalah firewall aplikasi web yang dapat Anda gunakan untuk mengamankan layanan App Runner Anda. Dengan daftar kontrol akses AWS WAF web (web ACLs), Anda dapat melindungi titik akhir layanan App Runner dari eksploitasi web umum dan bot yang tidak diinginkan.

ACL web memberi Anda kontrol halus atas semua permintaan web yang masuk ke layanan App Runner Anda. Anda dapat menentukan aturan dalam ACL web untuk mengizinkan, memblokir, atau memantau lalu lintas web, untuk memastikan bahwa hanya permintaan yang sah dan sah yang mencapai aplikasi web Anda dan APIs. Anda dapat menyesuaikan aturan ACL web berdasarkan kebutuhan bisnis dan keamanan spesifik Anda. Untuk mempelajari lebih lanjut tentang keamanan infrastruktur dan praktik terbaik untuk menerapkan jaringan ACLs, lihat Mengontrol lalu lintas jaringan di Panduan Pengguna Amazon VPC.

> **Important**
>
> Aturan IP sumber untuk layanan pribadi App Runner yang terkait dengan web WAF ACLs tidak mematuhi aturan berbasis IP. Ini karena saat ini kami tidak mendukung penerusan data IP sumber permintaan ke layanan pribadi App Runner yang terkait dengan WAF. Jika aplikasi App Runner Anda memerlukan aturan kontrol lalu lintas masuk IP/CIDR sumber, Anda harus menggunakan aturan grup keamanan untuk titik akhir pribadi, bukan web WAF. ACLs

Alur permintaan web yang masuk

Saat ACL AWS WAF web dikaitkan dengan layanan App Runner, permintaan web yang masuk akan melalui proses berikut:

1. App Runner meneruskan konten permintaan asal ke. AWS WAF
2. AWS WAF memeriksa permintaan dan membandingkan isinya dengan aturan yang Anda tentukan di ACL web Anda.
3. Berdasarkan pemeriksaannya, AWS WAF mengembalikan `allow` atau `block` respons ke App Runner.
   - Jika `allow` respons dikembalikan, App Runner meneruskan permintaan ke aplikasi Anda.
   - Jika `block` respons dikembalikan, App Runner memblokir permintaan agar tidak mencapai aplikasi web Anda. Ini meneruskan `block` respons dari AWS WAF ke aplikasi Anda.

> **Note**
>
> Secara default App Runner memblokir permintaan jika tidak ada respons yang dikembalikan. AWS WAF

Untuk informasi selengkapnya tentang AWS WAF web ACLs, lihat Daftar kontrol akses Web (web ACLs) di Panduan AWS WAF Pengembang.

> **Note**
>
> Anda membayar AWS WAF harga standar. Anda tidak dikenakan biaya tambahan untuk menggunakan AWS WAF web ACLs untuk layanan App Runner Anda. Untuk informasi selengkapnya tentang harga, lihat AWS WAF Harga.

Mengaitkan web WAF ACLs ke layanan App Runner Anda

Berikut ini adalah proses tingkat tinggi untuk mengaitkan ACL AWS WAF web dengan layanan App Runner Anda:

1. Buat ACL web di AWS WAF konsol. Untuk informasi selengkapnya, lihat Membuat ACL web di Panduan AWS WAF Pengembang.
2. Perbarui izin AWS Identity and Access Management (IAM) Anda untuk. AWS WAF Untuk informasi selengkapnya, lihat Izin.
3. Kaitkan ACL web dengan layanan App Runner menggunakan salah satu metode berikut:
   - Konsol App Runner: Kaitkan ACL web yang ada menggunakan konsol App Runner saat Anda membuat atau memperbarui layanan App Runner. Untuk petunjuk, lihat Mengelola AWS WAF web ACLs.
   - AWS WAF console: Kaitkan ACL web menggunakan AWS WAF konsol untuk layanan App Runner yang ada. Untuk informasi selengkapnya, lihat Mengaitkan atau memisahkan ACL web dengan sumber daya AWS di Panduan Pengembang.AWS WAF
   - AWS CLI: Kaitkan ACL web menggunakan AWS WAF publik APIs. Untuk informasi selengkapnya tentang AWS WAF publik APIs, lihat AssociateWebACL di Panduan Referensi AWS WAF API.

Pertimbangan

- Aturan IP sumber untuk layanan pribadi App Runner yang terkait dengan web WAF ACLs tidak mematuhi aturan berbasis IP. Ini karena saat ini kami tidak mendukung penerusan data IP sumber permintaan ke layanan pribadi App Runner yang terkait dengan WAF. Jika aplikasi App Runner Anda memerlukan aturan kontrol lalu lintas masuk IP/CIDR sumber, Anda harus menggunakan aturan grup keamanan untuk titik akhir pribadi, bukan web WAF. ACLs
- Layanan App Runner hanya dapat dikaitkan dengan satu ACL web. Namun, Anda dapat mengaitkan satu ACL web dengan beberapa layanan App Runner dan dengan beberapa AWS sumber daya. Contohnya termasuk kumpulan pengguna Amazon Cognito dan sumber daya Application Load Balancer.
- Saat Anda membuat ACL web, sejumlah kecil waktu berlalu sebelum ACL web sepenuhnya menyebar dan tersedia untuk App Runner. Waktu propagasi bisa dari beberapa detik hingga beberapa menit. AWS WAF mengembalikan a `WAFUnavailableEntityException` ketika Anda mencoba mengaitkan ACL web sebelum sepenuhnya disebarkan.

  Jika Anda menyegarkan browser atau menjauh dari konsol App Runner sebelum ACL web disebarkan sepenuhnya, asosiasi gagal terjadi. Namun, Anda dapat menavigasi di dalam konsol App Runner.
- AWS WAF mengembalikan `WAFNonexistentItemException` kesalahan saat Anda memanggil salah satu dari berikut ini AWS WAF APIs untuk layanan App Runner yang dalam keadaan tidak valid:
  - `AssociateWebACL`
  - `DisassociateWebACL`
  - `GetWebACLForResource`

  Status tidak valid untuk layanan App Runner Anda meliputi:
  - `CREATE_FAILED`
  - `DELETE_FAILED`
  - `DELETED`
  - `OPERATION_IN_PROGRESS`

> **Note**
>
> `OPERATION_IN_PROGRESS`status tidak valid hanya jika layanan App Runner Anda sedang dihapus.

- Permintaan Anda mungkin menghasilkan muatan yang lebih besar dari batas apa yang AWS WAF dapat diperiksa. Untuk informasi selengkapnya tentang cara AWS WAF menangani permintaan oversize dari App Runner, lihat Penanganan komponen permintaan Oversize di Panduan AWS WAF Pengembang untuk mempelajari cara AWS WAF menangani permintaan oversize dari App Runner.
- Jika Anda tidak menetapkan aturan yang sesuai atau pola lalu lintas Anda berubah, ACL web mungkin tidak seefektif mengamankan aplikasi Anda.

Izin

Untuk bekerja dengan ACL web AWS App Runner, tambahkan izin IAM berikut untuk: AWS WAF

- `apprunner:ListAssociatedServicesForWebAcl`
- `apprunner:DescribeWebAclForService`
- `apprunner:AssociateWebAcl`
- `apprunner:DisassociateWebAcl`

Untuk informasi selengkapnya tentang izin IAM, lihat Kebijakan dan izin di IAM di Panduan Pengguna IAM.

Berikut ini adalah contoh kebijakan IAM yang diperbarui untuk AWS WAF. Kebijakan IAM ini mencakup izin yang diperlukan untuk bekerja dengan layanan App Runner.

Example

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

```
            "Action":[
               "wafv2:ListResourcesForWebACL",
               "wafv2:GetWebACLForResource",
               "wafv2:AssociateWebACL",
               "wafv2:DisassociateWebACL",
               "apprunner:ListAssociatedServicesForWebAcl",
               "apprunner:DescribeWebAclForService",
               "apprunner:AssociateWebAcl",
               "apprunner:DisassociateWebAcl"
            ],
            "Resource":"*"
         }
      ]
}
```

> **Note**
>
> Meskipun Anda harus memberikan izin IAM, tindakan yang tercantum hanya izin dan tidak sesuai dengan operasi API.

Mengelola AWS WAF web ACLs

Mengelola AWS WAF web ACLs untuk layanan App Runner Anda dengan menggunakan salah satu metode berikut:

- the section called “Konsol Pelari Aplikasi”
- the section called “AWS CLI”

Konsol Pelari Aplikasi

Saat membuat layanan atau memperbarui layanan yang sudah ada di konsol App Runner, Anda dapat mengaitkan atau memisahkan ACL AWS WAF web.

> **Note**
>
> - Layanan App Runner hanya dapat dikaitkan dengan satu ACL web. Namun, Anda dapat mengaitkan satu ACL web dengan lebih dari satu layanan App Runner selain sumber daya lainnya AWS .

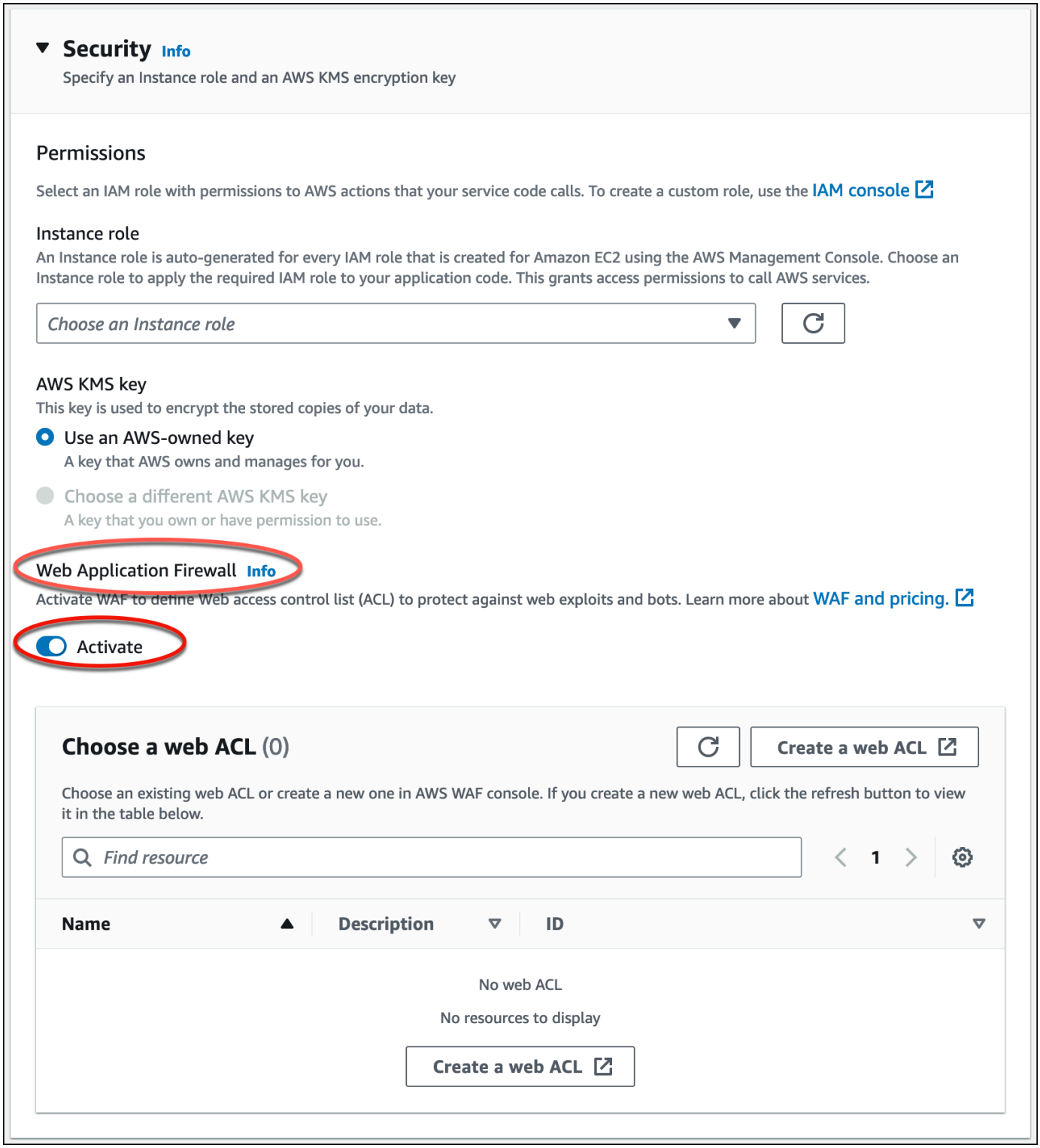
- Sebelum Anda mengaitkan ACL web, pastikan untuk memperbarui izin IAM Anda untuk. AWS WAF Untuk informasi selengkapnya, lihat Izin.

Mengaitkan AWS WAF web ACL

> **Important**
>
> Aturan IP sumber untuk layanan pribadi App Runner yang terkait dengan web WAF ACLs tidak mematuhi aturan berbasis IP. Ini karena saat ini kami tidak mendukung penerusan data IP sumber permintaan ke layanan pribadi App Runner yang terkait dengan WAF. Jika aplikasi App Runner Anda memerlukan aturan kontrol lalu lintas masuk IP/CIDR sumber, Anda harus menggunakan aturan grup keamanan untuk titik akhir pribadi, bukan web WAF. ACLs

Untuk mengaitkan ACL AWS WAF web

1. Buka konsol App Runner, dan di daftar Wilayah, pilih Wilayah AWS.
2. Berdasarkan apakah Anda membuat atau memperbarui layanan, lakukan salah satu langkah berikut:
   - Jika Anda membuat layanan baru, pilih Buat layanan Pelari Aplikasi dan buka Konfigurasi Layanan.
   - Jika Anda memperbarui layanan yang ada, pilih tab Konfigurasi, lalu pilih Edit di bawah Konfigurasi layanan.
3. Buka firewall aplikasi Web di bawah Keamanan.
4. Pilih tombol sakelar Aktifkan untuk melihat opsi.



5. Lakukan salah satu langkah berikut:

   - Untuk mengaitkan ACL web yang ada: Pilih ACL web yang diperlukan dari tabel Pilih ACL web untuk dikaitkan dengan layanan App Runner Anda.

- Untuk membuat ACL web baru: Pilih Buat ACL web untuk membuat ACL web baru menggunakan konsol. AWS WAF Untuk informasi selengkapnya, lihat Membuat ACL web di Panduan AWS WAF Pengembang.
  1. Pilih tombol refresh untuk melihat ACL web yang baru dibuat di tabel Choose a web ACL.
  2. Pilih ACL web yang diperlukan.

6. Pilih Berikutnya jika Anda membuat layanan baru atau Simpan perubahan jika Anda memperbarui layanan yang ada. ACL web yang dipilih dikaitkan dengan layanan App Runner Anda.
7. Untuk memverifikasi asosiasi ACL web, pilih tab Konfigurasi layanan Anda dan buka Konfigurasi layanan. Gulir ke firewall aplikasi Web di bawah Keamanan untuk melihat detail ACL web yang terkait dengan layanan Anda.

> **Note**
>
> Saat Anda membuat ACL web, sejumlah kecil waktu berlalu sebelum ACL web sepenuhnya menyebar dan tersedia untuk App Runner. Waktu propagasi bisa dari beberapa detik hingga beberapa menit. AWS WAF mengembalikan a `WAFUnavailableEntityException` ketika Anda mencoba mengaitkan ACL web sebelum sepenuhnya disebarkan.
> Jika Anda menyegarkan browser atau menjauh dari konsol App Runner sebelum ACL web disebarkan sepenuhnya, asosiasi gagal terjadi. Namun, Anda dapat menavigasi dalam konsol App Runner.

Memutuskan hubungan ACL web AWS WAF

Anda dapat memisahkan AWS WAF web ACl yang tidak lagi Anda perlukan dengan memperbarui layanan App Runner.

Untuk memisahkan web AWS WAF ACl

1. Buka konsol App Runner, dan di daftar Wilayah, pilih Wilayah AWS.
2. Buka tab Konfigurasi layanan yang ingin Anda perbarui dan pilih Edit di bawah Konfigurasi layanan.
3. Buka firewall aplikasi Web di bawah Keamanan.
4. Nonaktifkan tombol sakelar Aktifkan. Anda menerima pesan untuk mengonfirmasi penghapusan.

5. Pilih Konfirmasi. ACL web dipisahkan dari layanan App Runner Anda.

> **Note**
>
> - Jika Anda ingin mengaitkan layanan Anda dengan ACL web lain, pilih ACL web dari tabel Pilih ACL web. App Runner memisahkan ACL web saat ini dan memulai proses untuk mengaitkan dengan ACL web yang dipilih.
> - Jika tidak ada layanan atau sumber daya App Runner lain yang menggunakan ACL web yang tidak terkait, pertimbangkan untuk menghapus ACL web. Jika tidak, Anda akan terus mengeluarkan biaya. Untuk informasi lebih lanjut tentang harga, lihat AWS WAF Harga. Untuk instruksi tentang cara menghapus ACL web, lihat DeleteWebACL di Referensi AWS WAF API.
> - Anda tidak dapat menghapus ACL web yang terkait dengan layanan App Runner aktif lainnya atau sumber daya lainnya.

AWS CLI

Anda dapat mengaitkan atau memisahkan ACL AWS WAF web dengan menggunakan publik. AWS WAF APIs Layanan App Runner, yang dengannya Anda ingin mengaitkan atau memisahkan ACL web, harus dalam keadaan valid.

AWS WAF mengembalikan `WAFNonexistentItemException` kesalahan saat Anda memanggil salah satu dari berikut ini AWS WAF APIs untuk layanan App Runner yang dalam keadaan tidak valid:

- `AssociateWebACL`
- `DisassociateWebACL`
- `GetWebACLForResource`

Status tidak valid untuk layanan App Runner Anda meliputi:

- `CREATE_FAILED`
- `DELETE_FAILED`
- `DELETED`
- `OPERATION_IN_PROGRESS`

> **Note**
>
> `OPERATION_IN_PROGRESS`status tidak valid hanya jika layanan App Runner Anda sedang dihapus.

Untuk informasi selengkapnya tentang AWS WAF publik APIs, lihat Panduan Referensi AWS WAF API.

> **Note**
>
> Perbarui izin IAM Anda untuk. AWS WAF Untuk informasi selengkapnya, lihat Izin.

Mengaitkan AWS WAF web ACL menggunakan AWS CLI

> **Important**
>
> Aturan IP sumber untuk layanan pribadi App Runner yang terkait dengan web WAF ACLs tidak mematuhi aturan berbasis IP. Ini karena saat ini kami tidak mendukung penerusan data IP sumber permintaan ke layanan pribadi App Runner yang terkait dengan WAF. Jika aplikasi App Runner Anda memerlukan aturan kontrol lalu lintas masuk IP/CIDR sumber, Anda harus menggunakan aturan grup keamanan untuk titik akhir pribadi, bukan web WAF. ACLs

Untuk mengaitkan ACL AWS WAF web

1. Buat ACL AWS WAF web untuk layanan Anda dengan serangkaian tindakan aturan pilihan Anda `Allow` atau `Block` permintaan web ke layanan Anda. Untuk informasi selengkapnya AWS WAF APIs, lihat CreateWebACL di Panduan Referensi AWS WAF API.

   Example Buat ACL web - Permintaan

   ```
   aws wafv2
   create-web-acl
   --region <region>
   --name <web-acl-name>
   --scope REGIONAL
   ```

```
--default-action Allow={}
--visibility-config <file-name.json>
# This is the file containing the WAF web ACL rules.
```

2. Kaitkan ACL web yang Anda buat dengan layanan App Runner menggunakan API `associate-web-acl` AWS WAF publik. Untuk informasi selengkapnya AWS WAF APIs, lihat AssociateWebACL di Panduan Referensi AWS WAF API.

> **Note**
>
> Saat Anda membuat ACL web, sejumlah kecil waktu berlalu sebelum ACL web sepenuhnya menyebar dan tersedia untuk App Runner. Waktu propagasi bisa dari beberapa detik hingga beberapa menit. AWS WAF mengembalikan a `WAFUnavailableEntityException` ketika Anda mencoba mengaitkan ACL web sebelum sepenuhnya disebarkan.
> Jika Anda menyegarkan browser atau menjauh dari konsol App Runner sebelum ACL web disebarkan sepenuhnya, asosiasi gagal terjadi. Namun, Anda dapat menavigasi dalam konsol App Runner.

Example Mengaitkan ACL web - Permintaan

```
aws wafv2 associate-web-acl
--resource-arn <apprunner_service_arn>
--web-acl-arn <web_acl_arn>
--region <region>
```

3. Verifikasi bahwa ACL web dikaitkan dengan layanan App Runner Anda menggunakan API `get-web-acl-for-resource` AWS WAF publik. Untuk informasi selengkapnya AWS WAF APIs, lihat GetWebACLForSumber Daya di Panduan Referensi AWS WAF API.

Example Verifikasi ACL web untuk sumber daya - Permintaan

```
aws wafv2 get-web-acl-for-resource
--resource-arn <apprunner_service_arn>
--region <region>
```

Jika tidak ada web ACLs yang terkait dengan layanan Anda, Anda menerima respons kosong.

Menghapus ACL AWS WAF web menggunakan AWS CLI

Anda tidak dapat menghapus ACL AWS WAF web jika dikaitkan dengan layanan App Runner.

Untuk menghapus ACL AWS WAF web

1. Putuskan hubungan ACL web dari layanan App Runner Anda dengan menggunakan API publik. `disassociate-web-acl` AWS WAF Untuk informasi selengkapnya AWS WAF APIs, lihat DisassociateWebACL di Panduan Referensi AWS WAF API.

   Example Memutuskan hubungan ACL web - Permintaan

   ```
   aws wafv2 disassociate-web-acl
   --resource-arn <apprunner_service_arn>
   --region <region>
   ```

2. Verifikasi bahwa ACL web terputus dari layanan App Runner Anda menggunakan API publik. `get-web-acl-for-resource` AWS WAF

   Example Verifikasi bahwa ACL web tidak terkait - Permintaan

   ```
   aws wafv2 get-web-acl-for-resource
   --resource-arn <apprunner_service_arn>
   --region <region>
   ```

   ACL web yang tidak terkait tidak terdaftar untuk layanan App Runner Anda. Jika tidak ada web ACLs yang terkait dengan layanan Anda, Anda menerima respons kosong.

3. Hapus ACL web yang tidak terkait menggunakan API `delete-web-acl` AWS WAF publik. Untuk informasi selengkapnya AWS WAF APIs, lihat DeleteWebACL di Panduan Referensi AWS WAF API.

   Example Hapus ACL web - Permintaan

   ```
   aws wafv2 delete-web-acl
   --name <web_acl_name>
   --scope REGIONAL
   --id <web_acl_id>
   --lock-token <web_acl_lock_token>
   --region <region>
   ```

4. Verifikasi bahwa ACL web dihapus menggunakan API `list-web-acl` AWS WAF publik. Untuk informasi selengkapnya AWS WAF APIs, lihat ListWebACLsdi Panduan Referensi AWS WAF API.

   Example Verifikasi bahwa ACL web dihapus - Permintaan

   ```
   aws wafv2 list-web-acls
   --scope REGIONAL
   --region <region>
   ```

   ACL web yang dihapus tidak lagi terdaftar.

   > **Note**
   > Jika ACL web dikaitkan dengan layanan App Runner aktif lainnya atau sumber daya lainnya, seperti kumpulan pengguna Amazon Cognito, ACL web tidak dapat dihapus.

Daftar layanan App Runner yang terkait dengan ACL web

ACL web dapat dikaitkan dengan beberapa layanan App Runner dan sumber daya lainnya. Buat daftar layanan App Runner yang terkait dengan ACL web menggunakan API `list-resources-for-web-acl` AWS WAF publik. Untuk informasi selengkapnya AWS WAF APIs, lihat ListResourcesForWebACL di Panduan Referensi AWS WAF API.

Example Daftar layanan App Runner yang terkait dengan ACL web - Permintaan

```
aws wafv2 list-resources-for-web-acl
--web-acl-arn <WEB_ACL_ARN>
--resource-type APP_RUNNER_SERVICE
--region <REGION>
```

Example Daftar layanan App Runner yang terkait dengan ACL web - Response

Contoh berikut mengilustrasikan respons ketika tidak ada layanan App Runner yang terkait dengan ACL web.

```
{
```

```
  "ResourceArns": []
}
```

Example Daftar layanan App Runner yang terkait dengan ACL web - Response

Contoh berikut mengilustrasikan respons ketika ada layanan App Runner yang terkait dengan ACL web.

```
{
  "ResourceArns": [
    "arn:aws:apprunner:<region>:<aws_account_id>:service/<service_name>/<service_id>"
  ]
}
```

Menguji dan mencatat AWS WAF web ACLs

Saat Anda menetapkan tindakan aturan ke Count di ACL web Anda, AWS WAF tambahkan permintaan ke hitungan permintaan yang cocok dengan aturan. Untuk menguji ACL web dengan layanan App Runner Anda, tetapkan tindakan aturan ke Hitung dan pertimbangkan volume permintaan yang cocok dengan setiap aturan. Misalnya, Anda menetapkan aturan untuk `Block` tindakan yang cocok dengan sejumlah besar permintaan yang Anda tentukan sebagai lalu lintas pengguna normal. Dalam hal ini, Anda mungkin perlu mengkonfigurasi ulang aturan Anda. Untuk informasi selengkapnya, lihat Menguji dan menyetel AWS WAF perlindungan Anda di Panduan AWS WAF Pengembang.

Anda juga dapat mengonfigurasi header permintaan log AWS WAF ke grup CloudWatch log Amazon Logs, bucket Amazon Simple Storage Service (Amazon S3), atau Amazon Data Firehose. Untuk informasi selengkapnya, lihat Mencatat lalu lintas ACL web di Panduan AWS WAF Pengembang.

Untuk mengakses log yang terkait dengan ACL web yang terkait dengan layanan App Runner Anda, lihat bidang log berikut:

- `httpSourceName`: Berisi `APPRUNNER`
- `httpSourceId`: Berisi `customeraccountid-apprunnerserviceid`

Untuk informasi selengkapnya, lihat Contoh Log di Panduan AWS WAF Pengembang.

> **Important**
>
> Aturan IP sumber untuk layanan pribadi App Runner yang terkait dengan web WAF ACLs tidak mematuhi aturan berbasis IP. Ini karena saat ini kami tidak mendukung penerusan data IP sumber permintaan ke layanan pribadi App Runner yang terkait dengan WAF. Jika aplikasi App Runner Anda memerlukan aturan kontrol lalu lintas masuk IP/CIDR sumber, Anda harus menggunakan aturan grup keamanan untuk titik akhir pribadi, bukan web WAF. ACLs

Menyetel opsi layanan App Runner menggunakan file konfigurasi

> **Note**
>
> File konfigurasi hanya berlaku untuk layanan yang didasarkan pada kode sumber. Anda tidak dapat menggunakan file konfigurasi dengan layanan berbasis gambar.

Saat Anda membuat AWS App Runner layanan menggunakan repositori kode sumber, AWS App Runner memerlukan informasi tentang membangun dan memulai layanan Anda. Anda dapat memberikan informasi ini setiap kali membuat layanan menggunakan konsol App Runner atau API. Atau, Anda dapat mengatur opsi layanan dengan menggunakan file konfigurasi. Opsi yang Anda tentukan dalam file menjadi bagian dari repositori sumber Anda, dan setiap perubahan pada opsi ini dilacak mirip dengan bagaimana perubahan pada kode sumber dilacak. Anda dapat menggunakan file konfigurasi App Runner untuk menentukan lebih banyak opsi daripada yang didukung API. Anda tidak perlu menyediakan file konfigurasi jika Anda hanya memerlukan opsi dasar yang didukung API.

File konfigurasi App Runner adalah file YAMAL yang diberi nama `apprunner.yaml` di direktori sumber repositori aplikasi Anda. Ini menyediakan opsi build dan runtime untuk layanan Anda. Nilai dalam file ini menginstruksikan App Runner cara membuat dan memulai layanan Anda, dan menyediakan konteks runtime seperti pengaturan jaringan dan variabel lingkungan.

File konfigurasi App Runner tidak menyertakan pengaturan operasional, seperti CPU dan memori.

Untuk contoh file konfigurasi App Runner, lihatthe section called “Contoh”. Untuk panduan referensi lengkap, lihatthe section called “Referensi”.

Topik

- Contoh file konfigurasi App Runner
- Referensi file konfigurasi App Runner

Contoh file konfigurasi App Runner

> **Note**
>
> File konfigurasi hanya berlaku untuk layanan yang didasarkan pada kode sumber. Anda tidak dapat menggunakan file konfigurasi dengan layanan berbasis gambar.

Contoh berikut menunjukkan file AWS App Runner konfigurasi. Beberapa minimal dan hanya berisi pengaturan yang diperlukan. Lainnya lengkap, termasuk semua bagian file konfigurasi. Untuk ikhtisar file konfigurasi App Runner, lihatFile konfigurasi App Runner.

Contoh file konfigurasi

File konfigurasi minimal

Dengan file konfigurasi minimal, App Runner membuat asumsi berikut:

- Tidak ada variabel lingkungan khusus yang diperlukan selama membangun atau menjalankan.
- Versi runtime terbaru digunakan.
- Nomor port default dan variabel lingkungan port digunakan.

Example apprunner.yaml

```
version: 1.0
runtime: python3
build:
  commands:
    build:
      - pip install pipenv
      - pipenv install
run:
  command: python app.py
```

File konfigurasi lengkap

Contoh ini menunjukkan penggunaan semua kunci konfigurasi dalam format `apprunner.yaml` asli dengan runtime terkelola.

Example apprunner.yaml

```
version: 1.0
runtime: python3
build:
  commands:
    pre-build:
      - wget -c https://s3.amazonaws.com/amzn-s3-demo-bucket/test-lib.tar.gz -O - | tar
 -xz
    build:
      - pip install pipenv
      - pipenv install
    post-build:
      - python manage.py test
  env:
    - name: DJANGO_SETTINGS_MODULE
      value: "django_apprunner.settings"
    - name: MY_VAR_EXAMPLE
      value: "example"
run:
  runtime-version: 3.7.7
  command: pipenv run gunicorn django_apprunner.wsgi --log-file -
  network:
    port: 8000
    env: MY_APP_PORT
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
  secrets:
    - name: my-secret
      value-from: "arn:aws:secretsmanager:us-
east-1:123456789012:secret:testingstackAppRunnerConstr-kJFXde2ULKbT-S7t8xR:username::"
    - name: my-parameter
      value-from: "arn:aws:ssm:us-east-1:123456789012:parameter/parameter-name"
    - name: my-parameter-only-name
      value-from: "parameter-name"
```

File konfigurasi lengkap — (menggunakan build yang direvisi)

Contoh ini menunjukkan penggunaan semua kunci konfigurasi di `apprunner.yaml` dengan runtime terkelola.

`pre-run`Parameter ini hanya didukung oleh build App Runner yang direvisi. Jangan masukkan parameter ini dalam file konfigurasi jika aplikasi Anda menggunakan versi runtime yang didukung oleh build App Runner asli. Untuk informasi selengkapnya, lihat Versi runtime terkelola dan build App Runner.

> **Note**
>
> Karena contoh ini untuk Python 3.11, kita menggunakan perintah dan`pip3`. `python3` Untuk informasi selengkapnya, lihat Callout untuk versi runtime tertentu di topik platform Python.

Example apprunner.yaml

```
version: 1.0
runtime: python311
build:
  commands:
    pre-build:
      - wget -c https://s3.amazonaws.com/amzn-s3-demo-bucket/test-lib.tar.gz -O - | tar
 -xz
    build:
      - pip3 install pipenv
      - pipenv install
    post-build:
      - python3 manage.py test
  env:
    - name: DJANGO_SETTINGS_MODULE
      value: "django_apprunner.settings"
    - name: MY_VAR_EXAMPLE
      value: "example"
run:
  runtime-version: 3.11
  pre-run:
    - pip3 install pipenv
    - pipenv install
    - python3 copy-global-files.py
  command: pipenv run gunicorn django_apprunner.wsgi --log-file -
  network:
    port: 8000
    env: MY_APP_PORT
  env:
    - name: MY_VAR_EXAMPLE
```

```
      value: "example"
  secrets:
    - name: my-secret
      value-from: "arn:aws:secretsmanager:us-
east-1:123456789012:secret:testingstackAppRunnerConstr-kJFXde2ULKbT-S7t8xR:username::"
    - name: my-parameter
      value-from: "arn:aws:ssm:us-east-1:123456789012:parameter/parameter-name"
    - name: my-parameter-only-name
      value-from: "parameter-name"
```

Untuk contoh file konfigurasi runtime terkelola tertentu, lihat subtopik runtime tertentu di bawah. Layanan berbasis kode

Referensi file konfigurasi App Runner

> **Note**
>
> File konfigurasi hanya berlaku untuk layanan yang didasarkan pada kode sumber. Anda tidak dapat menggunakan file konfigurasi dengan layanan berbasis gambar.

Topik ini adalah panduan referensi komprehensif untuk sintaks dan semantik file konfigurasi. AWS App Runner Untuk ikhtisar file konfigurasi App Runner, lihatFile konfigurasi App Runner.

File konfigurasi App Runner adalah file YAMM. Beri nama`apprunner.yaml`, dan letakkan di direktori sumber repositori aplikasi Anda.

Ikhtisar struktur

File konfigurasi App Runner adalah file YAMM. Beri nama`apprunner.yaml`, dan letakkan di direktori sumber repositori aplikasi Anda.

File konfigurasi App Runner berisi bagian-bagian utama ini:

- Bagian atas - Berisi tombol tingkat atas
- Bagian Build - Mengonfigurasi tahap build
- Jalankan bagian - Mengonfigurasi tahap runtime

Bagian atas

Kunci di bagian atas file memberikan informasi umum tentang file dan runtime layanan Anda. Kunci-kunci berikut tersedia:

- `version`— Diperlukan. Versi file konfigurasi App Runner. Idealnya, gunakan versi terbaru.

  Sintaksis

  ```
  version: version
  ```

  Example

  ```
  version: 1.0
  ```

- `runtime`— Diperlukan. Nama runtime yang digunakan aplikasi Anda. Untuk mempelajari tentang runtime yang tersedia untuk berbagai platform pemrograman yang ditawarkan App Runner, lihat. Layanan berbasis kode

  > **Note**
  >
  > Konvensi penamaan dari runtime terkelola adalah`<language-name><major-version>`.

  Sintaksis

  ```
  runtime: runtime-name
  ```

  Example

  ```
  runtime: python3
  ```

Membangun bagian

Bagian build mengonfigurasi tahap pembuatan penerapan layanan App Runner. Anda dapat menentukan perintah build dan variabel lingkungan. Perintah build diperlukan.

Bagian dimulai dengan `build:` kunci, dan memiliki subkunci berikut:

- `commands`— Diperlukan. Menentukan perintah yang dijalankan App Runner selama berbagai fase build. Termasuk subkunci berikut:
  - `pre-build`— Opsional. Perintah yang dijalankan App Runner sebelum build. Misalnya, instal npm dependensi atau pustaka uji.
  - `build`— Diperlukan. Perintah yang dijalankan App Runner untuk membangun aplikasi Anda. Misalnya, gunakanpipenv.
  - `post-build`— Opsional. Perintah yang dijalankan App Runner setelah build. Misalnya, gunakan Maven untuk mengemas artefak build ke dalam file JAR atau WAR, atau jalankan pengujian.

  Sintaksis

  ```
  build:
    commands:
      pre-build:
        - command
        - …
      build:
        - command
        - …
      post-build:
        - command
        - …
  ```

  Example

  ```
  build:
    commands:
      pre-build:
        - yum install openssl
      build:
        - pip install -r requirements.txt
      post-build:
        - python manage.py test
  ```

- `env`— Opsional. Menentukan variabel lingkungan kustom untuk tahap build. Didefinisikan sebagai pemetaan skalar nama-nilai. Anda dapat merujuk ke variabel-variabel ini berdasarkan nama dalam perintah build Anda.

> **Note**
>
> Ada dua `env` entri berbeda di dua lokasi berbeda dalam file konfigurasi ini. Satu set ada di bagian Build dan yang lainnya di bagian Run.
>
> - `envSet` di bagian Build dapat direferensikan oleh `pre-run` perintah `pre-buildbuild`,`post-build`,, dan selama proses build.
>
> Penting - Perhatikan bahwa `pre-run` perintah terletak di bagian Jalankan file ini, meskipun mereka hanya dapat mengakses variabel lingkungan yang didefinisikan di bagian Build.
>
> - `envSet` di bagian Run dapat direferensikan oleh `run` perintah di lingkungan runtime.

Sintaksis

```
build:
  env:
    - name: name1
      value: value1
    - name: name2
      value: value2
    - …
```

Example

```
build:
  env:
    - name: DJANGO_SETTINGS_MODULE
      value: "django_apprunner.settings"
    - name: MY_VAR_EXAMPLE
      value: "example"
```

Jalankan bagian

Bagian run mengonfigurasi tahap proses container dari penerapan aplikasi App Runner. Anda dapat menentukan versi runtime, perintah pra-jalankan (hanya format yang direvisi), perintah mulai, port jaringan, dan variabel lingkungan.

Bagian dimulai dengan `run:` kunci, dan memiliki subkunci berikut:

- `runtime-version`— Opsional. Menentukan versi runtime yang ingin dikunci untuk layanan App Runner.

  Secara default, hanya versi utama yang terkunci. App Runner menggunakan versi minor dan patch terbaru yang tersedia untuk runtime pada setiap penerapan atau pembaruan layanan. Jika Anda menentukan versi mayor dan minor, keduanya menjadi terkunci, dan App Runner hanya memperbarui versi tambalan. Jika Anda menentukan versi mayor, minor, dan patch, layanan Anda dikunci pada versi runtime tertentu dan App Runner tidak pernah memperbaruinya.

  Sintaksis

  ```
  run:
    runtime-version: major[.minor[.patch]]
  ```

  > **Note**
  >
  > Runtime dari beberapa platform memiliki komponen versi yang berbeda. Lihat topik platform tertentu untuk detailnya.

  Example

  ```
  runtime: python3
  run:
    runtime-version: 3.7
  ```

- `pre-run`— Opsional. Hanya penggunaan build yang direvisi. Menentukan perintah yang dijalankan App Runner setelah menyalin aplikasi Anda dari image build ke image run. Anda dapat memasukkan perintah di sini untuk memodifikasi gambar run di luar `/app` direktori. Misalnya, jika Anda perlu menginstal dependensi global tambahan yang berada di luar `/app` direktori, masukkan perintah yang diperlukan di sub-bagian ini untuk melakukannya. Untuk informasi selengkapnya tentang proses pembuatan App Runner, lihatVersi runtime terkelola dan build App Runner.

  > **Note**
  >
  > - Penting — Meskipun `pre-run` perintah tercantum di bagian Run, mereka hanya dapat mereferensikan variabel lingkungan yang ditentukan di bagian Build dari file konfigurasi

ini. Mereka tidak dapat mereferensikan variabel lingkungan yang ditentukan dalam bagian Jalankan ini.

- `pre-run`Parameter ini hanya didukung oleh build App Runner yang direvisi. Jangan masukkan parameter ini dalam file konfigurasi jika aplikasi Anda menggunakan versi runtime yang didukung oleh build App Runner asli. Untuk informasi selengkapnya, lihat Versi runtime terkelola dan build App Runner.

Sintaksis

```
run:
  pre-run:
      - command
      - …
```

- `command`— Diperlukan. Perintah yang digunakan App Runner untuk menjalankan aplikasi Anda setelah menyelesaikan pembuatan aplikasi.

  Sintaksis

```
run:
  command: command
```

- `network`— Opsional. Menentukan port yang didengarkan aplikasi Anda. Ini termasuk yang berikut:
  - `port`— Opsional. Jika ditentukan, ini adalah nomor port yang didengarkan aplikasi Anda. Default-nya adalah `8080`.
  - `env`— Opsional. Jika ditentukan, App Runner meneruskan nomor port ke container dalam variabel lingkungan ini, selain (bukan sebagai ganti) meneruskan nomor port yang sama dalam variabel lingkungan default,`PORT`. Dengan kata lain, jika Anda menentukan`env`, App Runner meneruskan nomor port dalam dua variabel lingkungan.

  Sintaksis

```
run:
  network:
    port: port-number
    env: env-variable-name
```

Example

```
run:
  network:
    port: 8000
    env: MY_APP_PORT
```

- `env`— Opsional. Definisi variabel lingkungan kustom untuk tahap run. Didefinisikan sebagai pemetaan skalar nama-nilai. Anda dapat merujuk ke variabel-variabel ini dengan nama di lingkungan runtime Anda.

> **Note**
>
> Ada dua `env` entri berbeda di dua lokasi berbeda dalam file konfigurasi ini. Satu set ada di bagian Build dan yang lainnya di bagian Run.
>
> - `envSet` di bagian Build dapat direferensikan oleh `pre-run` perintah `pre-buildbuild`,`post-build`,, dan selama proses build.
>
> Penting - Perhatikan bahwa `pre-run` perintah terletak di bagian Jalankan file ini, meskipun mereka hanya dapat mengakses variabel lingkungan yang didefinisikan di bagian Build.
>
> - `envSet` di bagian Run dapat direferensikan oleh `run` perintah di lingkungan runtime.

Sintaksis

```
run:
  env:
    - name: name1
      value: value1
    - name: name2
      value: value2
  secrets:
    - name: name1
      value-from: arn:aws:secretsmanager:region:aws_account_id:secret:secret-id
    - name: name2
      value-from: arn:aws:ssm:region:aws_account_id:parameter/parameter-name
    - …
```

Example

```
run:
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
  secrets:
    - name: my-secret
      value-from: "arn:aws:secretsmanager:us-
east-1:123456789012:secret:testingstackAppRunnerConstr-kJFXde2ULKbT-
S7t8xR:username::"
    - name: my-parameter
      value-from: "arn:aws:ssm:us-east-1:123456789012:parameter/parameter-name"
    - name: my-parameter-only-name
      value-from: "parameter-name"
```

API Pelari Aplikasi

AWS App Runner Application programming interface (API) adalah RESTful API untuk membuat permintaan ke layanan App Runner. Anda dapat menggunakan API untuk membuat, membuat daftar, mendeskripsikan, memperbarui, dan menghapus resource App Runner di aplikasi Anda Akun AWS.

Anda dapat memanggil API secara langsung di kode aplikasi Anda, atau Anda dapat menggunakan salah satunya AWS SDKs.

Untuk informasi referensi API selengkapnya, lihat Referensi AWS App Runner API.

Untuk informasi selengkapnya tentang alat AWS pengembang, lihat Alat untuk Dibangun AWS.

Topik

- Menggunakan AWS CLI untuk bekerja dengan App Runner
- Menggunakan AWS CloudShell untuk bekerja dengan AWS App Runner

Menggunakan AWS CLI untuk bekerja dengan App Runner

Untuk skrip baris perintah, gunakan tombol AWS CLIuntuk melakukan panggilan ke layanan App Runner. Untuk informasi AWS CLI referensi lengkap, lihat apprunner di AWS CLI Command Reference.

AWS CloudShell memungkinkan Anda untuk melewatkan menginstal AWS CLI di lingkungan pengembangan Anda, dan menggunakannya AWS Management Console sebagai gantinya. Selain menghindari instalasi, Anda juga tidak perlu mengonfigurasi kredensi, dan Anda tidak perlu menentukan wilayah. AWS Management Console Sesi Anda memberikan konteks ini ke AWS CLI. Untuk informasi selengkapnya tentang CloudShell, dan untuk contoh penggunaan, lihatthe section called “Menggunakan AWS CloudShell”.

Menggunakan AWS CloudShell untuk bekerja dengan AWS App Runner

AWS CloudShell adalah shell berbasis browser dan pra-otentikasi yang dapat Anda luncurkan langsung dari file. AWS Management Console Anda dapat menjalankan AWS CLI perintah terhadap AWS layanan (termasuk AWS App Runner) menggunakan shell pilihan Anda (Bash, PowerShell atau Z shell). Anda juga dapat melakukan ini tanpa perlu mengunduh atau menginstal alat baris perintah.

Anda meluncurkan AWS CloudShell dari AWS Management Console, dan AWS kredenal yang Anda gunakan untuk masuk ke konsol secara otomatis tersedia di sesi shell baru. Pra-otentikasi AWS CloudShell pengguna ini memungkinkan Anda untuk melewati konfigurasi kredensional saat berinteraksi dengan AWS layanan seperti App Runner menggunakan AWS CLI versi 2 (pra-instal pada lingkungan komputasi shell).

Topik

- Memperoleh izin IAM untuk AWS CloudShell
- Berinteraksi dengan App Runner menggunakan AWS CloudShell
- Memverifikasi layanan App Runner Anda menggunakan AWS CloudShell

Memperoleh izin IAM untuk AWS CloudShell

Dengan menggunakan sumber daya manajemen akses yang disediakan oleh AWS Identity and Access Management, administrator dapat memberikan izin kepada pengguna IAM sehingga mereka dapat mengakses AWS CloudShell dan menggunakan fitur lingkungan.

Cara tercepat bagi administrator untuk memberikan akses ke pengguna adalah melalui kebijakan AWS terkelola. Kebijakan AWS terkelola adalah kebijakan mandiri yang dibuat dan dikelola oleh AWS. Kebijakan AWS terkelola berikut ini CloudShell dapat dilampirkan ke identitas IAM:

- `AWSCloudShellFullAccess`: Memberikan izin untuk menggunakan AWS CloudShell dengan akses penuh ke semua fitur.

Jika ingin membatasi cakupan tindakan yang dapat dilakukan oleh pengguna IAM AWS CloudShell, Anda dapat membuat kebijakan kustom yang menggunakan kebijakan `AWSCloudShellFullAccess` terkelola sebagai templat. Untuk informasi selengkapnya tentang membatasi tindakan yang tersedia bagi pengguna CloudShell, lihat Mengelola AWS CloudShell akses dan penggunaan dengan kebijakan IAM di Panduan AWS CloudShell Pengguna.

> **Note**
>
> Identitas IAM Anda juga memerlukan kebijakan yang memberikan izin untuk melakukan panggilan ke App Runner. Untuk informasi selengkapnya, lihat the section called “Pelari Aplikasi dan IAM”.

Berinteraksi dengan App Runner menggunakan AWS CloudShell

Setelah Anda meluncurkan AWS CloudShell dari AWS Management Console, Anda dapat segera mulai berinteraksi dengan App Runner menggunakan antarmuka baris perintah.

Dalam contoh berikut, Anda mengambil informasi tentang salah satu layanan App Runner Anda yang menggunakan in AWS CLI . CloudShell

> **Note**
>
> Saat menggunakan AWS CLI in AWS CloudShell, Anda tidak perlu mengunduh atau menginstal sumber daya tambahan apa pun. Selain itu, karena Anda sudah diautentikasi di dalam shell, Anda tidak perlu mengonfigurasi kredensional sebelum melakukan panggilan.

Example Mengambil informasi layanan App Runner menggunakan AWS CloudShell

1. Dari AWS Management Console, Anda dapat meluncurkan CloudShell dengan memilih opsi berikut yang tersedia di bilah navigasi:
   - Pilih CloudShell ikonnya.
   - Mulai mengetik **`cloudshell`** di kotak pencarian, lalu pilih CloudShellopsi saat Anda melihatnya di hasil pencarian.
2. Untuk mencantumkan semua layanan App Runner saat ini di AWS akun Anda di AWS Wilayah sesi konsol, masukkan perintah berikut di baris CloudShell perintah:

   ```
   $ aws apprunner list-services
   ```

   Output mencantumkan informasi ringkasan untuk layanan Anda.

   ```
   {
     "ServiceSummaryList": [
       {
         "ServiceName": "my-app-1",
         "ServiceId": "8fe1e10304f84fd2b0df550fe98a71fa",
         "ServiceArn": "arn:aws:apprunner:us-east-2:123456789012:service/my-
   app-1/8fe1e10304f84fd2b0df550fe98a71fa",
         "ServiceUrl": "psbqam834h.us-east-1.awsapprunner.com",
         "CreatedAt": "2020-11-20T19:05:25Z",
   ```

```
      "UpdatedAt": "2020-11-23T12:41:37Z",
      "Status": "RUNNING"
    },
    {
      "ServiceName": "my-app-2",
      "ServiceId": "ab8f94cfe29a460fb8760afd2ee87555",
      "ServiceArn": "arn:aws:apprunner:us-east-2:123456789012:service/my-app-2/
ab8f94cfe29a460fb8760afd2ee87555",
      "ServiceUrl": "e2m8rrrx33.us-east-1.awsapprunner.com",
      "CreatedAt": "2020-11-06T23:15:30Z",
      "UpdatedAt": "2020-11-23T13:21:22Z",
      "Status": "RUNNING"
    }
  ]
}
```

3. Untuk mendapatkan deskripsi rinci tentang layanan App Runner tertentu, masukkan perintah berikut di baris CloudShell perintah, menggunakan salah satu yang ARNs diambil pada langkah sebelumnya:

```
$ aws apprunner describe-service --service-arn arn:aws:apprunner:us-
east-2:123456789012:service/my-app-1/8fe1e10304f84fd2b0df550fe98a71fa
```

Output mencantumkan deskripsi rinci tentang layanan yang Anda tentukan.

```
{
  "Service": {
    "ServiceName": "my-app-1",
    "ServiceId": "8fe1e10304f84fd2b0df550fe98a71fa",
    "ServiceArn": "arn:aws:apprunner:us-east-2:123456789012:service/my-
app-1/8fe1e10304f84fd2b0df550fe98a71fa",
    "ServiceUrl": "psbqam834h.us-east-1.awsapprunner.com",
    "CreatedAt": "2020-11-20T19:05:25Z",
    "UpdatedAt": "2020-11-23T12:41:37Z",
    "Status": "RUNNING",
    "SourceConfiguration": {
      "CodeRepository": {
        "RepositoryUrl": "https://github.com/my-account/python-hello",
        "SourceCodeVersion": {
          "Type": "BRANCH",
          "Value": "main"
        },
```

```
        "CodeConfiguration": {
          "CodeConfigurationValues": {
            "BuildCommand": "[pip install -r requirements.txt]",
            "Port": "8080",
            "Runtime": "PYTHON_3",
            "RuntimeEnvironmentVariables": [
              {
                "NAME": "Jane"
              }
            ],
            "StartCommand": "python server.py"
          },
          "ConfigurationSource": "API"
        }
      },
      "AutoDeploymentsEnabled": true,
      "AuthenticationConfiguration": {
        "ConnectionArn": "arn:aws:apprunner:us-east-2:123456789012:connection/my-
github-connection/e7656250f67242d7819feade6800f59e"
      }
    },
    "InstanceConfiguration": {
      "CPU": "1 vCPU",
      "Memory": "3 GB"
    },
    "HealthCheckConfiguration": {
      "Protocol": "TCP",
      "Path": "/",
      "Interval": 10,
      "Timeout": 5,
      "HealthyThreshold": 1,
      "UnhealthyThreshold": 5
    },
    "AutoScalingConfigurationSummary": {
      "AutoScalingConfigurationArn": "arn:aws:apprunner:us-
east-2:123456789012:autoscalingconfiguration/
DefaultConfiguration/1/00000000000000000000000000000001",
      "AutoScalingConfigurationName": "DefaultConfiguration",
      "AutoScalingConfigurationRevision": 1
    }
  }
}
```

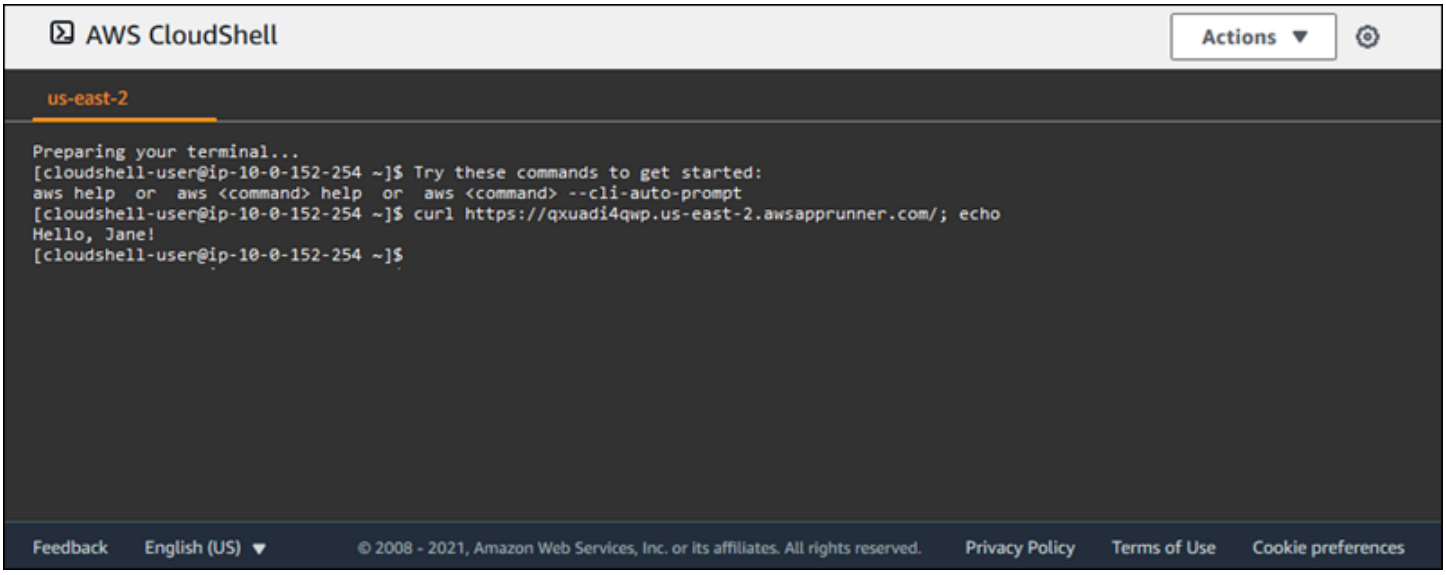
Memverifikasi layanan App Runner Anda menggunakan AWS CloudShell

Saat Anda membuat layanan App Runner, App Runner membuat domain default untuk situs web layanan Anda, dan menampilkannya di konsol (atau mengembalikannya dalam hasil panggilan API). Anda dapat menggunakan CloudShell untuk melakukan panggilan ke situs web Anda dan memverifikasi bahwa itu berfungsi dengan benar.

Misalnya, setelah Anda membuat layanan App Runner seperti yang dijelaskan dalamMemulai, jalankan perintah berikut di CloudShell:

```
$ curl https://qxuadi4qwp.us-east-2.awsapprunner.com/; echo
```

Output harus menampilkan konten halaman yang diharapkan.



Pemecahan Masalah

Bab ini menyediakan langkah-langkah pemecahan masalah untuk kesalahan umum dan masalah yang mungkin Anda temui saat menggunakan layanan Anda AWS App Runner . Pesan kesalahan dapat muncul di konsol, API, atau tab Log di halaman layanan Anda.

Untuk saran pemecahan masalah selengkapnya dan jawaban atas pertanyaan dukungan umum, kunjungi Pusat Pengetahuan .

Topik

- Ketika layanan gagal dibuat
- Nama domain kustom
- Kesalahan perutean permintaan HTTP/HTTPS
- Ketika layanan gagal terhubung ke Amazon RDS atau layanan hilir
- Ketika tidak ada cukup alamat IP untuk meluncurkan instance atau penskalaan

Ketika layanan gagal dibuat

Jika upaya Anda untuk membuat layanan App Runner gagal, layanan akan memasukkan `CREATE_FAILED` status. Status ini muncul sebagai Buat gagal di konsol. Layanan mungkin gagal dibuat karena masalah yang terkait dengan satu atau beberapa hal berikut:

- Kode aplikasi Anda
- Proses membangun
- Konfigurasi
- Kuota sumber daya
- Masalah sementara dengan dasar Layanan AWS yang digunakan layanan Anda

Untuk memecahkan masalah layanan yang gagal dibuat, kami sarankan Anda melakukan hal berikut.

1. Baca peristiwa dan log layanan untuk mengetahui apa yang menyebabkan layanan gagal dibuat.
2. Buat perubahan yang diperlukan pada kode atau konfigurasi Anda.
3. Jika Anda mencapai kuota layanan Anda, hapus satu atau beberapa layanan.

4. Jika Anda mencapai kuota sumber daya lain, Anda mungkin dapat meningkatkannya jika dapat disesuaikan.
5. Coba bangun kembali layanan setelah menyelesaikan semua langkah di atas. Untuk informasi tentang cara membangun kembali layanan Anda, lihatthe section called “Membangun kembali layanan yang gagal”.

> **Note**
>
> Salah satu kuota sumber daya yang dapat disesuaikan yang mungkin menyebabkan masalah adalah sumber daya vCPU Fargate On-Demand.
> Jumlah sumber daya vCPU menentukan jumlah instance yang dapat diberikan oleh App Runner ke layanan Anda. Ini adalah nilai kuota yang dapat disesuaikan untuk jumlah sumber daya vCPU Fargate On-Demand yang berada dalam layanan. AWS Fargate Untuk melihat pengaturan kuota vCPU untuk akun Anda atau meminta peningkatan kuota, gunakan konsol Service Quotas di. AWS Management Console Untuk informasi selengkapnya, lihat kuota AWS Fargate layanan di Panduan Pengembang Layanan Kontainer Elastis Amazon.

> **Important**
>
> Anda tidak dikenakan biaya tambahan di luar upaya pembuatan awal untuk layanan yang gagal. Meskipun layanan yang gagal tidak dapat digunakan, namun tetap diperhitungkan dalam kuota layanan Anda. App Runner tidak secara otomatis menghapus layanan yang gagal, jadi pastikan Anda menghapusnya setelah selesai menganalisis kegagalan.

Nama domain kustom

Bagian ini mencakup bagaimana Anda dapat memecahkan masalah dan menyelesaikan berbagai kesalahan yang mungkin Anda alami saat menautkan ke domain khusus.

> **Note**
>
> Untuk meningkatkan keamanan aplikasi App Runner Anda, domain*.awsapprunner.com terdaftar di Daftar Akhiran Publik (PSL). Untuk keamanan lebih lanjut, kami menyarankan Anda menggunakan cookie dengan `__Host-` awalan jika Anda perlu mengatur cookie

sensitif di nama domain default untuk aplikasi App Runner Anda. Praktik ini akan membantu mempertahankan domain Anda dari upaya pemalsuan permintaan lintas situs (CSRF). Untuk informasi selengkapnya, lihat halaman Set-Cookie di Jaringan Pengembang Mozilla.

Mendapatkan kesalahan Buat Gagal untuk domain kustom

- Periksa apakah kesalahan ini karena masalah dengan catatan CAA. Jika tidak ada catatan CAA di mana pun di pohon DNS, Anda menerima pesan`fail open`, dan AWS Certificate Manager mengeluarkan sertifikat untuk memverifikasi domain kustom. Hal ini memungkinkan App Runner untuk menerima domain kustom. Jika Anda menggunakan sertifikasi CAA dalam catatan DNS, pastikan setidaknya satu catatan CAA domain disertakan. `amazon.com` Jika tidak, ACM gagal mengeluarkan sertifikat. Akibatnya, domain kustom untuk App Runner gagal dibuat.

  Contoh berikut menggunakan alat pencarian DNS DiG untuk menunjukkan catatan CAA yang hilang entri yang diperlukan. Contoh digunakan `example.com` sebagai domain kustom. Jalankan perintah berikut dalam contoh untuk memeriksa catatan CAA.

  ```
  ...
  ;; QUESTION SECTION:
  ;example.com.           IN  CAA

  ;; ANSWER SECTION:
  example.com.        7200    IN  CAA 0 iodef "mailto:hostmaster@example.com"
  example.com.        7200    IN  CAA 0 issue "letsencrypt.org"
  ...note absence of "amazon.com" in any of the above CAA records...
  ```

- Perbaiki catatan domain dan pastikan bahwa setidaknya satu catatan CAA termasuk`amazon.com`.
- Coba lagi untuk menautkan domain kustom dengan App Runner.

Untuk petunjuk tentang cara mengatasi kesalahan CAA, lihat berikut ini:

- Masalah Otorisasi Otoritas Sertifikasi (CAA)
- Bagaimana cara mengatasi kesalahan CAA untuk menerbitkan atau memperbarui sertifikat ACM?

Mendapatkan validasi sertifikat DNS kesalahan tertunda untuk domain kustom

- Periksa apakah Anda melewatkan langkah penting dalam pengaturan domain khusus. Selain itu, periksa apakah Anda salah mengonfigurasi catatan DNS menggunakan alat pencarian DNS seperti DiG. Secara khusus, periksa kesalahan berikut:
  - Setiap langkah yang terlewatkan.
  - Karakter yang tidak didukung seperti kutipan ganda dalam catatan DNS.
- Perbaiki kesalahannya.
- Coba lagi untuk menautkan domain kustom dengan App Runner.

Untuk petunjuk tentang cara mengatasi kesalahan validasi CAA, lihat berikut ini.

- Validasi DNS
- the section called “Nama domain khusus”

Perintah pemecahan masalah dasar

- Konfirmasikan bahwa layanan dapat ditemukan.

```
aws apprunner list-services
```

- Jelaskan layanan dan periksa statusnya.

```
aws apprunner describe-service --service-arn
```

- Periksa status domain kustom.

```
aws apprunner describe-custom-domains --service-arn
```

- Buat daftar semua operasi yang sedang berlangsung.

```
aws apprunner list-operations --service-arn
```

Perpanjangan sertifikat domain kustom

Saat Anda menambahkan domain kustom ke layanan Anda, App Runner memberi Anda satu set catatan CNAME yang Anda tambahkan ke server DNS Anda. Catatan CNAME ini termasuk catatan sertifikat. App Runner menggunakan AWS Certificate Manager (ACM) untuk memverifikasi domain. App Runner memvalidasi catatan DNS ini untuk memastikan kepemilikan domain ini secara berkelanjutan. Jika Anda menghapus catatan CNAME dari zona DNS, App Runner tidak dapat lagi memvalidasi catatan DNS, dan sertifikat domain kustom gagal diperpanjang secara otomatis.

Bagian ini mencakup cara mengatasi masalah perpanjangan sertifikat domain kustom berikut:

- the section called “CNAME dihapus dari server DNS”.
- the section called “Sertifikat telah kedaluwarsa”.

CNAME dihapus dari server DNS

- Mengambil data CNAME Anda menggunakan DescribeCustomDomainsAPI atau dari setelan Domain Kustom di konsol App Runner. Untuk informasi tentang disimpan CNAMEs, lihat CertificateValidationRecords.
- Tambahkan catatan CNAME validasi sertifikat ke server DNS Anda. App Runner kemudian dapat memvalidasi bahwa Anda memiliki domain. Setelah Anda menambahkan catatan CNAME, dibutuhkan waktu hingga 30 menit agar catatan DNS disebarkan. Hal ini juga dapat mengambil beberapa jam untuk App Runner dan ACM untuk mencoba kembali proses perpanjangan sertifikat. Untuk petunjuk tentang cara menambahkan catatan CNAME, lihatthe section called “Mengelola domain kustom”.

Sertifikat telah kedaluwarsa

- Lepaskan (batalkan tautan) lalu kaitkan (tautkan) domain kustom untuk layanan App Runner Anda menggunakan konsol App Runner atau API. App Runner membuat catatan CNAME validasi sertifikat baru.

- Tambahkan catatan CNAME validasi sertifikat baru ke server DNS Anda.

  Untuk petunjuk tentang cara memisahkan (memutuskan tautan) dan mengaitkan (menautkan) domain kustom, lihat. the section called “Mengelola domain kustom”

Bagaimana cara memverifikasi bahwa sertifikat berhasil diperbarui

Anda dapat memeriksa status catatan sertifikat Anda untuk memverifikasi sertifikat Anda berhasil diperbarui. Anda dapat memeriksa status sertifikat dengan menggunakan alat seperti curl.

Untuk informasi selengkapnya tentang perpanjangan sertifikat, lihat tautan berikut:

- Mengapa sertifikat ACM saya ditandai sebagai tidak memenuhi syarat untuk perpanjangan?
- Perpanjangan terkelola untuk sertifikat ACM
- Validasi DNS

Kesalahan perutean permintaan HTTP/HTTPS

Bagian ini membahas bagaimana Anda dapat memecahkan masalah dan mengatasi kesalahan yang mungkin Anda alami saat merutekan lalu lintas HTTP/HTTPS ke titik akhir layanan App Runner Anda.

404 Kesalahan tidak ditemukan saat mengirim lalu lintas HTTP/HTTPS ke titik akhir layanan App Runner

- Verifikasi `Host Header` bahwa menunjuk ke URL layanan dalam permintaan HTTP karena App Runner menggunakan informasi header host untuk merutekan permintaan. Sebagian besar klien, seperti`cURL`, dan browser web secara otomatis mengarahkan header host ke URL layanan. Jika klien Anda tidak menyetel URL layanan sebagai`Host Header`, Anda menerima `404 Not Found` kesalahan.

  Example Header host salah

  ```
  $ ~ curl -I -H "host: foobar.com" https://testservice.awsapprunner.com/
  HTTP/1.1 404 Not Found
  transfer-encoding: chunked
  ```

Example Header host yang benar

```
$ ~ curl -I -H "host: testservice.awsapprunner.com" https://
testservice.awsapprunner.com/
HTTP/1.1 200 OK
content-length: 11772
content-type: text/html; charset=utf-8
```

- Verifikasi bahwa klien Anda mengatur indikator nama server (SNI) dengan benar untuk permintaan perutean ke layanan publik atau pribadi. Untuk penghentian TLS dan perutean permintaan, App Runner menggunakan set SNI dalam koneksi HTTPS.

Ketika layanan gagal terhubung ke Amazon RDS atau layanan hilir

Mungkin ada masalah konfigurasi jaringan dengan layanan Anda jika gagal terhubung ke database Amazon RDS atau aplikasi atau layanan hilir lainnya. Topik ini memandu Anda melalui beberapa langkah untuk menentukan apakah ada masalah dengan konfigurasi jaringan Anda dan opsi untuk memperbaikinya. Untuk mempelajari lebih lanjut tentang konfigurasi lalu lintas keluar untuk Pelari Aplikasi, lihat. Mengaktifkan akses VPC untuk lalu lintas keluar

> **Note**
>
> Untuk melihat konfigurasi Konektor VPC Anda, dari panel navigasi kiri konsol App Runner, pilih Konfigurasi jaringan. Kemudian pilih tab Lalu lintas keluar. Pilih Konektor VPC. Halaman berikutnya menampilkan detail tentang Konektor VPC. Dari halaman ini Anda dapat melihat dan menelusuri hal-hal berikut: Subnet, Grup keamanan, dan layanan App Runner yang menggunakan VPC.

Untuk mempersempit penyebab ketidakmampuan aplikasi Anda untuk terhubung ke layanan hilir lainnya

1. Pastikan subnet yang digunakan dalam Konektor VPC adalah subnet pribadi. Jika konektor dikonfigurasi dengan subnet publik, layanan Anda akan mengalami kesalahan, karena Hyperplane ENIs (antarmuka jaringan elastis) yang mendasarinya untuk setiap subnet tidak memiliki ruang IP publik.

Jika konektor VPC Anda menggunakan subnet publik, Anda memiliki opsi berikut untuk memperbaiki konfigurasi ini:

a. Buat subnet pribadi baru, dan gunakan sebagai pengganti subnet publik untuk Konektor VPC. Untuk informasi selengkapnya, lihat Subnet untuk VPC Anda di Panduan Pengguna Amazon VPC.

b. Rutekan subnet publik yang ada melalui gateway NAT. Untuk informasi selengkapnya, lihat gateway NAT di Panduan Pengguna Amazon VPC.

2. Verifikasi bahwa aturan masuk dan keluar grup keamanan untuk Konektor VPC sudah benar. Dari panel navigasi kiri konsol App Runner, pilih Konfigurasi jaringan > Lalu lintas keluar. Pilih Konektor VPC dari daftar. Halaman berikutnya mencantumkan grup Keamanan yang dapat Anda pilih untuk diperiksa.

3. Verifikasi bahwa aturan masuk dan keluar grup keamanan sudah benar untuk instans RDS atau layanan hilir lain yang Anda coba sambungkan. Untuk informasi selengkapnya, lihat panduan layanan untuk layanan hilir yang coba disambungkan oleh aplikasi App Runner Anda.

4. Untuk mengonfirmasi bahwa tidak ada jenis masalah penyiapan jaringan lain di luar konfigurasi App Runner Anda, coba sambungkan ke RDS atau layanan hilir di luar App Runner:

   a. Dari EC2 instance Amazon di VPC yang sama, coba sambungkan ke instans atau layanan RDS.

   b. Jika Anda mencoba menyambung ke titik akhir VPC layanan, verifikasi konektivitas dengan mengakses titik akhir yang sama dari instance EC2 di VPC yang sama.

5. Jika salah satu pengujian koneksi di Langkah 4 gagal, kemungkinan besar ada masalah di luar konfigurasi App Runner dengan sumber daya lain di akun Anda AWS. Hubungi AWS Support untuk bantuan untuk mengisolasi lebih lanjut dan memperbaiki masalah dengan konfigurasi jaringan Anda yang lain.

6. Jika Anda berhasil terhubung ke instans RDS atau layanan hilir dengan melakukan instruksi di Langkah 4, maka lanjutkan dengan instruksi di langkah ini. Kami akan memeriksa apakah lalu lintas memasuki ENI dengan mengaktifkan dan memeriksa log aliran ENI Hyperplane.

> **Note**
>
> Untuk dapat menyelesaikan langkah-langkah ini dan mendapatkan informasi log aliran ENI yang diperlukan, upaya koneksi ke RDS atau layanan hilir harus dilakukan setelah layanan App Runner Anda berhasil dijalankan. Aplikasi Anda harus melakukan operasi

sambungkan ke RDS atau layanan hilir saat berada dalam keadaan Running. Jika tidak, ENIs dapat dibersihkan sebagai bagian dari alur kerja rollback App Runner. Pendekatan ini memastikan bahwa ENIs tetap tersedia untuk penyelidikan lebih lanjut.

a. Dari AWS konsol, luncurkan EC2 konsol.

b. Dari panel navigasi kiri, dalam pengelompokan Jaringan & Keamanan, pilih Antarmuka Jaringan.

c. Gulir ke kolom Jenis Antarmuka dan Deskripsi untuk menemukan subnet yang terkait dengan Konektor VPC. ENIs Mereka akan memiliki pola penamaan berikut.

   - Jenis Antarmuka: fargate
   - Deskripsi: dimulai dengan AWSAppRunner ENI(contoh: AWSAppRunner ENI - abcde123-abcd-1234-1234-abcde1233456)

d. Gunakan kotak centang di awal baris untuk memilih ENIs yang berlaku.

e. Dari menu Tindakan pilih Buat log alur.

f. Masukkan informasi dalam petunjuk dan pilih Buat aliran flog di bagian bawah halaman.

g. Periksa log aliran yang dihasilkan.

   - Jika lalu lintas memasuki ENI saat Anda menguji koneksi, maka masalahnya tidak terkait dengan pengaturan ENI. Mungkin ada masalah konfigurasi jaringan dengan sumber daya lain di AWS Akun Anda selain layanan App Runner. Hubungi AWS Support untuk bantuan lebih lanjut.
   - Jika lalu lintas tidak memasuki ENI saat Anda menguji koneksi, kami sarankan Anda menghubungi AWS Support untuk melihat apakah ada masalah yang diketahui dengan layanan Fargate.

h. Gunakan alat Reachability Analyzer jaringan. Alat ini membantu menentukan kesalahan konfigurasi jaringan dengan mengidentifikasi komponen pemblokiran ketika sumber di jalur jaringan virtual tidak dapat dijangkau. Untuk informasi selengkapnya, lihat Apa itu Reachability Analyzer? di Panduan Penganalisis Reachability VPC Amazon.

   Masukkan App Runner ENI sebagai sumber, dan RDS ENI sebagai tujuan.

7. Jika Anda tidak dapat mempersempit masalah lebih lanjut, atau jika Anda masih tidak dapat terhubung ke RDS atau layanan hilir setelah menyelesaikan langkah-langkah sebelumnya, kami sarankan Anda menghubungi AWS Support untuk bantuan lebih lanjut.

Ketika tidak ada cukup alamat IP untuk meluncurkan instance atau penskalaan

> **Note**
>
> Untuk layanan publik, App Runner tidak membuat Antarmuka Jaringan Elastis (ENI) di Anda VPCs, sehingga layanan publik Anda tidak terpengaruh oleh perubahan ini.

Panduan ini membantu Anda mengatasi kesalahan kelelahan IP yang mungkin Anda temui pada layanan App Runner dengan akses VPC untuk lalu lintas keluar diaktifkan.

App Runner akan meluncurkan instance di subnet yang terkait dengan konektor VPC Anda. App Runner membuat 1 ENI per instance di subnet tempat instance Anda diluncurkan. Setiap ENI menggunakan IP pribadi di subnet itu. Subnet memiliki jumlah tetap yang IPs tersedia, tergantung pada blok CIDR yang terkait dengan subnet tersebut. Jika App Runner tidak dapat menemukan subnet dengan cukup IPs untuk membuat ENI, itu akan gagal meluncurkan instance baru untuk layanan App Runner Anda. Hal ini dapat menyebabkan masalah dengan meningkatkan layanan Anda. Dalam kasus seperti itu, Anda akan melihat log peristiwa App Runner yang menunjukkan bahwa App Runner tidak dapat menemukan subnet yang tersedia. IPs Anda dapat memperbarui layanan Anda dengan instruksi di bawah ini untuk mengatasi kesalahan tersebut.

Cara memperbarui layanan Anda agar lebih banyak tersedia IPs

Jumlah alamat IP yang tersedia di subnet didasarkan pada blok CIDR yang terkait dengan subnet tersebut. Blok CIDR yang terkait dengan subnet tidak dapat diperbarui setelah pembuatan. Konektor VPC App Runner juga tidak dapat diperbarui setelah dibuat. Untuk memberikan lebih banyak IPs ke layanan App Runner Anda dengan akses VPC untuk lalu lintas keluar diaktifkan:

1. Buat subnet baru dengan blok CIDR yang lebih besar.
2. Buat konektor VPC baru dengan subnet baru.
3. Perbarui layanan App Runner Anda untuk menggunakan konektor VPC baru.

Menghitung IPs yang diperlukan untuk layanan Anda

Sebelum mencoba membuat subnet baru dengan blok CIDR yang lebih besar, tentukan jumlah yang IPs Anda perlukan di seluruh layanan App Runner Anda. Sebaiknya hitung jumlah yang IPs dibutuhkan di konektor Anda sebagai berikut:

1. Untuk setiap layanan dengan akses VPC untuk lalu lintas keluar diaktifkan, perhatikan ukuran maksimal (instance maksimum) dalam konfigurasi penskalaan otomatis.
2. Jumlahkan nilai di semua layanan.
3. Gandakan jumlah ini untuk memperhitungkan instance baru yang diluncurkan selama penerapan biru-hijau.

Contoh

Pertimbangkan dua layanan A dan B menggunakan konektor VPC yang sama.

1. Layanan A memiliki ukuran maksimal yang dikonfigurasi sebagai 25.
2. Layanan B memiliki ukuran maksimal yang dikonfigurasi sebagai 15.

Diperlukan IPs = 2 × (25 + 15) = 80

Pastikan subnet Anda memiliki setidaknya 80 IPs kombinasi yang tersedia.

Buat subnet baru

1. Tentukan ukuran blok CIDR yang diperlukan untuk IPv4 menggunakan rumus ini (Perhatikan bahwa 5 IPs dicadangkan oleh AWS: Subnet Sizing)

```
Number of available IP addresses = 2^(32 - prefix length) - 5
```

```
Example :
For 192.168.1.0/24:
Prefix length is 24
Number of available IP addresses = 2^(32 - 24) - 5 = 2^8-5 = 251 IP addresses

For 10.0.0.0/16:
Prefix length is 16
Number of available IP addresses = 2^(32 - 16) - 5 = 2^16-5 = 65,531 IP addresses
```

```
Quick reference:
/24 = 251 IP addresses
/16 = 65,531 IP addresses
```

2. Buat subnet baru dengan menggunakan AWS EC2 CLI.

```
aws ec2 create-subnet --vpc-id <my-vpc-id> --cidr-block <cidr-block>
```

Contoh (membuat subnet dengan 4.096 IPs):

```
aws ec2 create-subnet --vpc-id my-vpc-id --cidr-block 10.0.0.0/20
```

3. Buat konektor VPC baru. Lihat: Mengelola Akses VPC
4. Perbarui layanan Anda dengan lalu lintas keluar ke VPC yang diaktifkan untuk menggunakan konektor VPC baru ini. App Runner akan mulai menggunakan subnet baru setelah layanan Anda diperbarui.

> **Note**
>
> VPCs juga terbatas dengan jumlah IPs yang tersedia yang dapat dialokasikan ke subnet oleh blok CIDR. Jika Anda tidak dapat membuat subnet dengan blok CIDR yang lebih besar, Anda mungkin perlu memperbarui VPC Anda dengan blok CIDR sekunder sebelum membuat subnet baru.

Melampirkan blok CIDR Sekunder ke VPC Anda

Kaitkan blok CIDR sekunder ke VPC ini.

```
aws ec2 associate-vpc-cidr-block --vpc-id <my-vpc-id> --cidr-block <cidr-block>
```

Contoh:

```
aws ec2 associate-vpc-cidr-block --vpc-id my-vpc-id --cidr-block 10.1.0.0/16
```

Verifikasi

Setelah Anda memperbarui layanan Anda. Anda dapat menggunakan yang berikut ini untuk melakukan verifikasi perbaikan

1. Pantau log peristiwa: Pantau log peristiwa layanan App Runner Anda untuk memvalidasi tidak ada IP baru atau kesalahan ketidaktersediaan ENI yang muncul
2. Periksa Penskalaan Layanan:
    1. Tingkatkan layanan sepenuhnya dengan mengubah jumlah instans min dalam konfigurasi penskalaan otomatis Anda
    2. Verifikasi bahwa semua instance baru diluncurkan tanpa kesalahan terkait IP
    3. Memantau melalui beberapa peristiwa penskalaan untuk memastikan kinerja yang konsisten
3. Spanduk Konsol: Jika Anda menggunakan AWS Management Console, konfirmasikan bahwa App Runner tidak lagi menampilkan peringatan spanduk tentang tidak mencukupi IPs.
4. Pemanfaatan VPC dan Subnet IP:
    1. Gunakan Dashboard VPC atau perintah CLI untuk memeriksa penggunaan alamat IP di subnet baru Anda.
    2. Konfirmasikan bahwa masih ada margin yang sehat yang tersedia IPs setelah layanan Anda ditingkatkan

Perangkap Umum

Saat mengatasi kelelahan IP di layanan App Runner, perhatikan potensi masalah ini:

1. Perencanaan Alamat IP yang Tidak Memadai: Meremehkan kebutuhan IP masa depan dapat menyebabkan masalah kelelahan berulang. Melakukan perencanaan kapasitas secara menyeluruh, dengan mempertimbangkan potensi pertumbuhan layanan dan skenario penggunaan puncak.
2. Mengabaikan Penggunaan IP VPC: Ingatlah bahwa layanan AWS lain dalam VPC yang sama juga menggunakan alamat IP. Pertimbangkan persyaratan IP dari semua layanan saat merencanakan konfigurasi VPC dan subnet Anda.
3. Mengabaikan Update Services: Setelah membuat subnet baru atau konektor VPC, pastikan Anda memperbarui layanan App Runner untuk menggunakan konfigurasi baru. Kegagalan untuk melakukannya akan mengakibatkan penggunaan berkelanjutan dari rentang IP yang habis.

4. Kesalahpahaman Blok CIDR Tumpang tindih: Saat menambahkan blok CIDR sekunder ke VPC, pastikan blok tersebut tidak tumpang tindih dengan blok yang ada. Blok CIDR yang tumpang tindih dapat menyebabkan konflik perutean dan ambiguitas alamat IP.
5. Melebihi Batas VPC: Ketahuilah bahwa VPC dapat memiliki maksimum 5 blok CIDR (1 primer dan 4 sekunder). Rencanakan ekspansi ruang alamat IP Anda dalam batasan ini.
6. Mengabaikan Distribusi Subnet AZ: Saat membuat subnet baru, pastikan subnet didistribusikan di beberapa Availability Zone untuk ketersediaan tinggi dan toleransi kesalahan.
7. Mengabaikan Batas ENI: Ingatlah bahwa ada batasan jumlah ENIs yang dapat dilampirkan ke instance. Verifikasi bahwa batas akun AWS Anda selaras dengan penggunaan antarmuka jaringan yang Anda rencanakan.

Dengan mengetahui jebakan ini, Anda dapat mengelola sumber daya VPC dengan lebih efektif dan menghindari masalah kelelahan IP di layanan App Runner Anda.

Sumber Daya Tambahan

1. Dokumentasi AWS VPC
2. Memahami Blok CIDR
3. Konektor VPC Pelari Aplikasi

Glosarium

1. ENI: Antarmuka Jaringan Elastis, antarmuka jaringan virtual di AWS.
2. CIDR: Classless Inter-Domain Routing, metode untuk mengalokasikan alamat IP.
3. Konektor VPC: Sumber daya yang memungkinkan App Runner terhubung ke VPC Anda.

Keamanan di App Runner

Keamanan cloud di AWS adalah prioritas tertinggi. Sebagai AWS pelanggan, Anda mendapat manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan.

Keamanan adalah tanggung jawab bersama antara Anda AWS dan Anda. Model tanggung jawab bersama menjelaskan hal ini sebagai keamanan cloud dan keamanan dalam cloud:

- Keamanan cloud — AWS bertanggung jawab untuk melindungi infrastruktur yang menjalankan AWS layanan di AWS Cloud. AWS juga memberi Anda layanan yang dapat Anda gunakan dengan aman. Auditor pihak ketiga secara teratur menguji dan memverifikasi efektivitas keamanan kami sebagai bagian dari Program AWS Kepatuhan Program AWS Kepatuhan . Untuk mempelajari tentang program kepatuhan yang berlaku AWS App Runner, lihat AWS Layanan dalam Lingkup oleh AWS Layanan Program Kepatuhan .
- Keamanan di cloud — Tanggung jawab Anda ditentukan oleh AWS layanan yang Anda gunakan. Anda juga bertanggung jawab atas faktor lain, yang mencakup sensitivitas data Anda, persyaratan perusahaan Anda, serta undang-undang dan peraturan yang berlaku.

Dokumentasi ini membantu Anda memahami cara menerapkan model tanggung jawab bersama saat menggunakan App Runner. Topik berikut menunjukkan cara mengonfigurasi App Runner untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga mempelajari cara menggunakan AWS layanan lain yang membantu Anda memantau dan mengamankan sumber daya App Runner Anda.

Topik

- Perlindungan data di App Runner
- Manajemen identitas dan akses untuk App Runner
- Pencatatan dan pemantauan di App Runner
- Validasi kepatuhan untuk App Runner
- Ketahanan di App Runner
- Keamanan infrastruktur di AWS App Runner
- Menggunakan App Runner dengan titik akhir VPC
- Analisis konfigurasi dan kerentanan di App Runner
- Praktik terbaik keamanan untuk App Runner

Perlindungan data di App Runner

Model tanggung jawab AWS bersama model berlaku untuk perlindungan data di AWS App Runner. 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 kredensyal 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 Runner 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.

Untuk topik keamanan App Runner lainnya, lihatKeamanan.

Topik

- Melindungi data menggunakan enkripsi
- Privasi lalu lintas antar jaringan

Melindungi data menggunakan enkripsi

AWS App Runner membaca sumber aplikasi Anda (gambar sumber atau kode sumber) dari repositori yang Anda tentukan dan menyimpannya untuk penyebaran ke layanan Anda. Untuk informasi selengkapnya, lihat Arsitektur dan konsep.

Perlindungan data mengacu pada melindungi data saat transit (saat bepergian ke dan dari App Runner) dan saat istirahat (saat disimpan di pusat AWS data).

Untuk informasi selengkapnya tentang perlindungan data, lihatthe section called “Perlindungan data”.

Untuk topik keamanan App Runner lainnya, lihatKeamanan.

Enkripsi bergerak

Anda dapat mencapai perlindungan data dalam perjalanan dengan dua cara: mengenkripsi koneksi menggunakan Transport Layer Security (TLS), atau menggunakan enkripsi sisi klien (di mana objek dienkripsi sebelum dikirim). Kedua metode ini berlaku untuk melindungi data aplikasi Anda. Untuk mengamankan koneksi, enkripsi menggunakan TLS setiap kali aplikasi Anda, pengembang dan administratornya, dan pengguna akhirnya mengirim atau menerima objek apa pun. App Runner menyiapkan aplikasi Anda untuk menerima lalu lintas melalui TLS.

Enkripsi sisi klien bukanlah metode yang valid untuk melindungi gambar sumber atau kode yang Anda berikan kepada App Runner untuk penerapan. App Runner memerlukan akses ke sumber aplikasi Anda, sehingga tidak dapat dienkripsi. Oleh karena itu, pastikan untuk mengamankan koneksi antara lingkungan pengembangan atau penyebaran Anda dan App Runner.

Enkripsi saat istirahat dan manajemen kunci

Untuk melindungi data aplikasi Anda saat istirahat, App Runner mengenkripsi semua salinan yang disimpan dari gambar sumber aplikasi atau bundel sumber Anda. Saat Anda membuat layanan App Runner, Anda dapat memberikan AWS KMS key. Jika Anda menyediakannya, App Runner menggunakan kunci yang Anda berikan untuk mengenkripsi sumber Anda. Jika Anda tidak menyediakannya, App Runner menggunakan sebagai Kunci yang dikelola AWS gantinya.

Untuk detail tentang parameter pembuatan layanan App Runner, lihat CreateService. Untuk informasi tentang AWS Key Management Service (AWS KMS), lihat Panduan AWS Key Management Service Pengembang.

Privasi lalu lintas antar jaringan

App Runner menggunakan Amazon Virtual Private Cloud (Amazon VPC) untuk membuat batasan antara sumber daya di aplikasi App Runner dan mengontrol lalu lintas di antara sumber daya, jaringan lokal, dan internet. Untuk informasi selengkapnya tentang keamanan VPC Amazon, lihat Privasi lalu lintas Internetwork di Amazon VPC di Panduan Pengguna Amazon VPC.

Untuk informasi tentang mengaitkan aplikasi App Runner Anda dengan VPC Amazon khusus, lihat. the section called “Lalu lintas keluar”

Untuk informasi tentang mengamankan permintaan ke App Runner menggunakan titik akhir VPC, lihat. the section called “Titik akhir VPC”

Untuk informasi selengkapnya tentang perlindungan data, lihatthe section called “Perlindungan data”.

Untuk topik keamanan App Runner lainnya, lihatKeamanan.

Manajemen identitas dan akses untuk App Runner

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 Runner. IAM adalah Layanan AWS yang dapat Anda gunakan tanpa biaya tambahan.

Untuk topik keamanan App Runner lainnya, lihatKeamanan.

Topik

- Audiens
- Mengautentikasi dengan identitas
- Mengelola akses menggunakan kebijakan
- Bagaimana App Runner bekerja dengan IAM
- Contoh kebijakan berbasis identitas App Runner
- Menggunakan peran terkait layanan untuk App Runner
- AWS kebijakan terkelola untuk AWS App Runner
- Memecahkan masalah identitas dan akses App Runner

Audiens

Cara Anda menggunakan AWS Identity and Access Management (IAM) berbeda, tergantung pada pekerjaan yang Anda lakukan di App Runner.

Pengguna layanan — Jika Anda menggunakan layanan App Runner untuk melakukan pekerjaan Anda, administrator Anda memberi Anda kredensi dan izin yang Anda butuhkan. Saat Anda menggunakan lebih banyak fitur App Runner untuk melakukan pekerjaan Anda, Anda mungkin memerlukan izin tambahan. Memahami cara akses dikelola dapat membantu Anda meminta izin yang tepat dari administrator Anda. Jika Anda tidak dapat mengakses fitur di App Runner, lihatMemecahkan masalah identitas dan akses App Runner.

Administrator layanan — Jika Anda bertanggung jawab atas sumber daya App Runner di perusahaan Anda, Anda mungkin memiliki akses penuh ke App Runner. Tugas Anda adalah menentukan fitur dan sumber daya App Runner mana yang harus diakses pengguna layanan Anda. Kemudian, Anda harus mengirimkan permintaan kepada administrator IAM untuk mengubah izin pengguna layanan Anda. Tinjau informasi di halaman ini untuk memahami konsep dasar IAM. Untuk mempelajari lebih lanjut tentang cara perusahaan Anda dapat menggunakan IAM dengan App Runner, lihat. Bagaimana App Runner bekerja dengan IAM

Administrator IAM — Jika Anda seorang administrator IAM, Anda mungkin ingin mempelajari detail tentang cara menulis kebijakan untuk mengelola akses ke App Runner. Untuk melihat contoh kebijakan berbasis identitas Pelari Aplikasi yang dapat Anda gunakan di IAM, lihat. Contoh kebijakan berbasis identitas App Runner

Mengautentikasi dengan identitas

Otentikasi adalah cara Anda masuk AWS menggunakan kredensi identitas Anda. Anda harus diautentikasi (masuk ke AWS) sebagai Pengguna root akun AWS, sebagai pengguna IAM, atau dengan mengasumsikan peran IAM.

Anda dapat masuk AWS sebagai identitas federasi dengan menggunakan kredensi yang disediakan melalui sumber identitas. AWS IAM Identity Center Pengguna (IAM Identity Center), autentikasi masuk tunggal perusahaan Anda, dan kredensi Google atau Facebook Anda adalah contoh identitas federasi. Saat Anda masuk sebagai identitas terfederasi, administrator Anda sebelumnya menyiapkan federasi identitas menggunakan peran IAM. Ketika Anda mengakses AWS dengan menggunakan federasi, Anda secara tidak langsung mengambil peran.

Bergantung pada jenis pengguna Anda, Anda dapat masuk ke AWS Management Console atau portal AWS akses. Untuk informasi selengkapnya tentang masuk AWS, lihat Cara masuk ke Panduan AWS Sign-In Pengguna Anda Akun AWS.

Jika Anda mengakses AWS secara terprogram, AWS sediakan kit pengembangan perangkat lunak (SDK) dan antarmuka baris perintah (CLI) untuk menandatangani permintaan Anda secara kriptografis dengan menggunakan kredensil Anda. Jika Anda tidak menggunakan AWS alat, Anda harus menandatangani permintaan sendiri. Guna mengetahui informasi selengkapnya tentang penggunaan metode yang disarankan untuk menandatangani permintaan sendiri, lihat AWS Signature Version 4 untuk permintaan API dalam Panduan Pengguna IAM.

Apa pun metode autentikasi yang digunakan, Anda mungkin diminta untuk menyediakan informasi keamanan tambahan. Misalnya, AWS merekomendasikan agar Anda menggunakan otentikasi multi-faktor (MFA) untuk meningkatkan keamanan akun Anda. Untuk mempelajari selengkapnya, lihat Autentikasi multi-faktor dalam Panduan Pengguna AWS IAM Identity Center dan Autentikasi multi-faktor AWS di IAM dalam Panduan Pengguna IAM.

Akun AWS pengguna root

Saat Anda membuat Akun AWS, Anda mulai dengan satu identitas masuk yang memiliki akses lengkap ke semua Layanan AWS dan sumber daya di akun. Identitas ini disebut pengguna Akun AWS root dan diakses dengan masuk dengan alamat email dan kata sandi yang Anda gunakan untuk membuat akun. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari. Lindungi kredensial pengguna root Anda dan gunakan kredensial tersebut untuk melakukan tugas yang hanya dapat dilakukan pengguna root. Untuk daftar lengkap tugas

yang mengharuskan Anda masuk sebagai pengguna root, lihat Tugas yang memerlukan kredensial pengguna root dalam Panduan Pengguna IAM.

Pengguna dan grup IAM

Pengguna IAM adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus untuk satu orang atau aplikasi. Jika memungkinkan, kami merekomendasikan untuk mengandalkan kredensial sementara, bukan membuat pengguna IAM yang memiliki kredensial jangka panjang seperti kata sandi dan kunci akses. Namun, jika Anda memiliki kasus penggunaan tertentu yang memerlukan kredensial jangka panjang dengan pengguna IAM, kami merekomendasikan Anda merotasi kunci akses. Untuk informasi selengkapnya, lihat Merotasi kunci akses secara teratur untuk kasus penggunaan yang memerlukan kredensial jangka panjang dalam Panduan Pengguna IAM.

Grup IAM adalah identitas yang menentukan sekumpulan pengguna IAM. Anda tidak dapat masuk sebagai grup. Anda dapat menggunakan grup untuk menentukan izin bagi beberapa pengguna sekaligus. Grup mempermudah manajemen izin untuk sejumlah besar pengguna sekaligus. Misalnya, Anda dapat meminta kelompok untuk menyebutkan IAMAdmins dan memberikan izin kepada grup tersebut untuk mengelola sumber daya IAM.

Pengguna berbeda dari peran. Pengguna secara unik terkait dengan satu orang atau aplikasi, tetapi peran dimaksudkan untuk dapat digunakan oleh siapa pun yang membutuhkannya. Pengguna memiliki kredensial jangka panjang permanen, tetapi peran memberikan kredensial sementara. Untuk mempelajari selengkapnya, lihat Kasus penggunaan untuk pengguna IAM dalam Panduan Pengguna IAM.

Peran IAM

Peran IAM adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus. Peran ini mirip dengan pengguna IAM, tetapi tidak terkait dengan orang tertentu. Untuk mengambil peran IAM sementara AWS Management Console, Anda dapat beralih dari pengguna ke peran IAM (konsol). Anda dapat mengambil peran dengan memanggil operasi AWS CLI atau AWS API atau dengan menggunakan URL kustom. Untuk informasi selengkapnya tentang cara menggunakan peran, lihat Metode untuk mengambil peran dalam Panduan Pengguna IAM.

Peran IAM dengan kredensial sementara berguna dalam situasi berikut:

- Akses pengguna terfederasi – Untuk menetapkan izin ke identitas terfederasi, Anda membuat peran dan menentukan izin untuk peran tersebut. Ketika identitas terfederasi mengautentikasi, identitas tersebut terhubung dengan peran dan diberi izin yang ditentukan oleh peran. Untuk informasi tentang peran untuk federasi, lihat Buat peran untuk penyedia identitas pihak

ketiga dalam Panduan Pengguna IAM. Jika menggunakan Pusat Identitas IAM, Anda harus mengonfigurasi set izin. Untuk mengontrol apa yang dapat diakses identitas Anda setelah identitas tersebut diautentikasi, Pusat Identitas IAM akan mengorelasikan set izin ke peran dalam IAM. Untuk informasi tentang set izin, lihat Set izin dalam Panduan Pengguna AWS IAM Identity Center .

- Izin pengguna IAM sementara – Pengguna atau peran IAM dapat mengambil peran IAM guna mendapatkan berbagai izin secara sementara untuk tugas tertentu.
- Akses lintas akun – Anda dapat menggunakan peran IAM untuk mengizinkan seseorang (prinsipal tepercaya) di akun lain untuk mengakses sumber daya di akun Anda. Peran adalah cara utama untuk memberikan akses lintas akun. Namun, dengan beberapa Layanan AWS, Anda dapat melampirkan kebijakan secara langsung ke sumber daya (alih-alih menggunakan peran sebagai proxy). Untuk mempelajari perbedaan antara peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Akses sumber daya lintas akun di IAM dalam Panduan Pengguna IAM.
- Akses lintas layanan — Beberapa Layanan AWS menggunakan fitur lain Layanan AWS. Misalnya, saat Anda melakukan panggilan dalam suatu layanan, biasanya layanan tersebut menjalankan aplikasi di Amazon EC2 atau menyimpan objek di Amazon S3. Sebuah layanan mungkin melakukannya menggunakan izin prinsipal yang memanggil, menggunakan peran layanan, atau peran terkait layanan.
  - Sesi akses teruskan (FAS) — Saat Anda menggunakan pengguna atau peran IAM untuk melakukan tindakan AWS, Anda dianggap sebagai prinsipal. Ketika Anda menggunakan beberapa layanan, Anda mungkin melakukan sebuah tindakan yang kemudian menginisiasi tindakan lain di layanan yang berbeda. FAS menggunakan izin dari pemanggilan utama Layanan AWS, dikombinasikan dengan permintaan Layanan AWS untuk membuat permintaan ke layanan hilir. Permintaan FAS hanya dibuat ketika layanan menerima permintaan yang memerlukan interaksi dengan orang lain Layanan AWS atau sumber daya untuk menyelesaikannya. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan ketika mengajukan permintaan FAS, lihat Sesi akses maju.
  - Peran layanan – Peran layanan adalah peran IAM yang dijalankan oleh layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, mengubah, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat Buat sebuah peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan pengguna IAM.
  - Peran terkait layanan — Peran terkait layanan adalah jenis peran layanan yang ditautkan ke peran layanan. Layanan AWS Layanan tersebut dapat menjalankan peran untuk melakukan tindakan atas nama Anda. Peran terkait layanan muncul di Anda Akun AWS dan dimiliki oleh layanan. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.

- Aplikasi yang berjalan di Amazon EC2 — Anda dapat menggunakan peran IAM untuk mengelola kredensi sementara untuk aplikasi yang berjalan pada EC2 instance dan membuat AWS CLI atau AWS permintaan API. Ini lebih baik untuk menyimpan kunci akses dalam EC2 instance. Untuk menetapkan AWS peran ke EC2 instance dan membuatnya tersedia untuk semua aplikasinya, Anda membuat profil instans yang dilampirkan ke instance. Profil instance berisi peran dan memungkinkan program yang berjalan pada EC2 instance untuk mendapatkan kredensi sementara. Untuk informasi selengkapnya, lihat Menggunakan peran IAM untuk memberikan izin ke aplikasi yang berjalan di EC2 instans Amazon di Panduan Pengguna IAM.

Mengelola akses menggunakan kebijakan

Anda mengontrol akses AWS dengan membuat kebijakan dan melampirkannya ke AWS identitas atau sumber daya. Kebijakan adalah objek AWS yang, ketika dikaitkan dengan identitas atau sumber daya, menentukan izinnya. AWS mengevaluasi kebijakan ini ketika prinsipal (pengguna, pengguna root, atau sesi peran) membuat permintaan. Izin dalam kebijakan menentukan apakah permintaan diizinkan atau ditolak. Sebagian besar kebijakan disimpan AWS sebagai dokumen JSON. Untuk informasi selengkapnya tentang struktur dan isi dokumen kebijakan JSON, lihat Gambaran umum kebijakan JSON dalam Panduan Pengguna IAM.

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Yaitu, principal dapat melakukan tindakan pada suatu sumber daya, dan dalam suatu syarat.

Secara default, pengguna dan peran tidak memiliki izin. Untuk memberikan izin kepada pengguna untuk melakukan tindakan di sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM. Administrator kemudian dapat menambahkan kebijakan IAM ke peran, dan pengguna dapat mengambil peran.

Kebijakan IAM mendefinisikan izin untuk suatu tindakan terlepas dari metode yang Anda gunakan untuk melakukan operasinya. Misalnya, anggaplah Anda memiliki kebijakan yang mengizinkan tindakan `iam:GetRole`. Pengguna dengan kebijakan tersebut bisa mendapatkan informasi peran dari AWS Management Console, API AWS CLI, atau AWS API.

Kebijakan berbasis identitas

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke sebuah identitas, seperti pengguna IAM, grup pengguna IAM, atau peran IAM. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan oleh pengguna dan peran, di sumber daya mana,

dan berdasarkan kondisi seperti apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Tentukan izin IAM kustom dengan kebijakan terkelola pelanggan dalam Panduan Pengguna IAM.

Kebijakan berbasis identitas dapat dikategorikan lebih lanjut sebagai kebijakan inline atau kebijakan yang dikelola. Kebijakan inline disematkan langsung ke satu pengguna, grup, atau peran. Kebijakan terkelola adalah kebijakan mandiri yang dapat Anda lampirkan ke beberapa pengguna, grup, dan peran dalam. Akun AWS Kebijakan AWS terkelola mencakup kebijakan terkelola dan kebijakan yang dikelola pelanggan. Untuk mempelajari cara memilih antara kebijakan yang dikelola atau kebijakan inline, lihat Pilih antara kebijakan yang dikelola dan kebijakan inline dalam Panduan Pengguna IAM.

Kebijakan berbasis sumber daya

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan kepercayaan peran IAM dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Untuk sumber daya tempat kebijakan dilampirkan, kebijakan menentukan tindakan apa yang dapat dilakukan oleh prinsipal tertentu pada sumber daya tersebut dan dalam kondisi apa. Anda harus menentukan prinsipal dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup akun, pengguna, peran, pengguna federasi, atau. Layanan AWS

Kebijakan berbasis sumber daya merupakan kebijakan inline yang terletak di layanan tersebut. Anda tidak dapat menggunakan kebijakan AWS terkelola dari IAM dalam kebijakan berbasis sumber daya.

Daftar kontrol akses (ACLs)

Access control lists (ACLs) mengontrol prinsipal mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACLs mirip dengan kebijakan berbasis sumber daya, meskipun mereka tidak menggunakan format dokumen kebijakan JSON.

Amazon S3, AWS WAF, dan Amazon VPC adalah contoh layanan yang mendukung. ACLs Untuk mempelajari selengkapnya ACLs, lihat Ringkasan daftar kontrol akses (ACL) di Panduan Pengembang Layanan Penyimpanan Sederhana Amazon.

Jenis-jenis kebijakan lain

AWS mendukung jenis kebijakan tambahan yang kurang umum. Jenis-jenis kebijakan ini dapat mengatur izin maksimum yang diberikan kepada Anda oleh jenis kebijakan yang lebih umum.

- Batasan izin – Batasan izin adalah fitur lanjutan tempat Anda mengatur izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas ke entitas IAM (pengguna IAM atau peran IAM). Anda dapat menetapkan batasan izin untuk suatu entitas. Izin yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas milik entitas dan batasan izinnya. Kebijakan berbasis sumber daya yang menentukan pengguna atau peran dalam bidang `Principal` tidak dibatasi oleh batasan izin. Penolakan eksplisit dalam salah satu kebijakan ini akan menggantikan pemberian izin. Untuk informasi selengkapnya tentang batasan izin, lihat Batasan izin untuk entitas IAM dalam Panduan Pengguna IAM.
- Kebijakan kontrol layanan (SCPs) — SCPs adalah kebijakan JSON yang menentukan izin maksimum untuk organisasi atau unit organisasi (OU) di. AWS Organizations AWS Organizations adalah layanan untuk mengelompokkan dan mengelola secara terpusat beberapa Akun AWS yang dimiliki bisnis Anda. Jika Anda mengaktifkan semua fitur dalam suatu organisasi, maka Anda dapat menerapkan kebijakan kontrol layanan (SCPs) ke salah satu atau semua akun Anda. SCP membatasi izin untuk entitas di akun anggota, termasuk masing-masing. Pengguna root akun AWS Untuk informasi selengkapnya tentang Organizations dan SCPs, lihat Kebijakan kontrol layanan di Panduan AWS Organizations Pengguna.
- Kebijakan kontrol sumber daya (RCPs) — RCPs adalah kebijakan JSON yang dapat Anda gunakan untuk menetapkan izin maksimum yang tersedia untuk sumber daya di akun Anda tanpa memperbarui kebijakan IAM yang dilampirkan ke setiap sumber daya yang Anda miliki. RCP membatasi izin untuk sumber daya di akun anggota dan dapat memengaruhi izin efektif untuk identitas, termasuk Pengguna root akun AWS, terlepas dari apakah itu milik organisasi Anda. Untuk informasi selengkapnya tentang Organizations dan RCPs, termasuk daftar dukungan Layanan AWS tersebut RCPs, lihat Kebijakan kontrol sumber daya (RCPs) di Panduan AWS Organizations Pengguna.
- Kebijakan sesi – Kebijakan sesi adalah kebijakan lanjutan yang Anda berikan sebagai parameter ketika Anda membuat sesi sementara secara programatis untuk peran atau pengguna terfederasi. Izin sesi yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas pengguna atau peran dan kebijakan sesi. Izin juga bisa datang dari kebijakan berbasis sumber daya. Penolakan eksplisit dalam salah satu kebijakan ini akan menggantikan pemberian izin. Untuk informasi selengkapnya, lihat Kebijakan sesi dalam Panduan Pengguna IAM.

Berbagai jenis kebijakan

Ketika beberapa jenis kebijakan berlaku pada suatu permintaan, izin yang dihasilkan lebih rumit untuk dipahami. Untuk mempelajari cara AWS menentukan apakah akan mengizinkan permintaan saat beberapa jenis kebijakan terlibat, lihat Logika evaluasi kebijakan di Panduan Pengguna IAM.

Bagaimana App Runner bekerja dengan IAM

Sebelum Anda menggunakan IAM untuk mengelola akses AWS App Runner, Anda harus memahami fitur IAM apa yang tersedia untuk digunakan dengan App Runner. Untuk mendapatkan tampilan tingkat tinggi tentang cara kerja App Runner dan AWS layanan lainnya dengan IAM, lihat AWS Layanan yang Bekerja dengan IAM di Panduan Pengguna IAM.

Untuk topik keamanan App Runner lainnya, lihatKeamanan.

Topik

- Kebijakan berbasis identitas App Runner
- Kebijakan berbasis sumber daya App Runner
- Otorisasi berdasarkan tag App Runner
- Izin pengguna App Runner
- Peran IAM Pelari Aplikasi

Kebijakan berbasis identitas App Runner

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 Runner 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. Tindakan kebijakan biasanya memiliki nama yang sama dengan operasi AWS API terkait. Ada beberapa pengecualian, misalnya tindakan hanya izin yang tidak memiliki operasi API yang cocok. Ada juga beberapa operasi yang memerlukan beberapa tindakan dalam suatu kebijakan. Tindakan tambahan ini disebut tindakan dependen.

Sertakan tindakan dalam kebijakan untuk memberikan izin untuk melakukan operasi terkait.

Tindakan kebijakan di App Runner menggunakan awalan berikut sebelum tindakan:. `apprunner:` Misalnya, untuk memberikan izin kepada seseorang untuk menjalankan EC2 instance Amazon dengan operasi Amazon EC2 `RunInstances` API, Anda menyertakan `ec2:RunInstances` tindakan tersebut dalam kebijakan mereka. Pernyataan kebijakan harus memuat elemen `Action` atau `NotAction`. App Runner mendefinisikan serangkaian tindakannya sendiri yang menjelaskan tugas yang dapat Anda lakukan dengan layanan ini.

Untuk menetapkan beberapa tindakan dalam satu pernyataan, pisahkan dengan koma seperti berikut:

```
"Action": [
   "apprunner:CreateService",
   "apprunner:CreateConnection"
]
```

Anda dapat menentukan beberapa tindakan menggunakan wildcard (*). Sebagai contoh, untuk menentukan semua tindakan yang dimulai dengan kata `Describe`, sertakan tindakan berikut:

```
"Action": "apprunner:Describe*"
```

Untuk melihat daftar tindakan Pelari Aplikasi, lihat Tindakan yang ditentukan oleh AWS App Runner dalam Referensi Otorisasi Layanan.

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. Pernyataan harus menyertakan elemen `Resource` atau `NotResource`. Praktik terbaiknya, tentukan sumber daya menggunakan Amazon Resource Name (ARN). Anda dapat melakukan ini untuk tindakan yang mendukung jenis sumber daya tertentu, yang dikenal sebagai izin tingkat sumber daya.

Untuk tindakan yang tidak mendukung izin di tingkat sumber daya, misalnya operasi pencantuman, gunakan wildcard (*) untuk menunjukkan bahwa pernyataan tersebut berlaku untuk semua sumber daya.

```
"Resource": "*"
```

Sumber daya App Runner memiliki struktur ARN berikut:

```
arn:aws:apprunner:region:account-id:resource-type/resource-name[/resource-id]
```

Untuk informasi selengkapnya tentang format ARNs, lihat Amazon Resource Names (ARNs) dan Ruang Nama AWS Layanan di bagian. Referensi Umum AWS

Misalnya, untuk menentukan `my-service` layanan dalam pernyataan Anda, gunakan ARN berikut:

```
"Resource": "arn:aws:apprunner:us-east-1:123456789012:service/my-service"
```

Untuk menentukan semua layanan milik akun tertentu, gunakan wildcard (*):

```
"Resource": "arn:aws:apprunner:us-east-1:123456789012:service/*"
```

Beberapa tindakan App Runner, seperti untuk membuat sumber daya, tidak dapat dilakukan pada sumber daya tertentu. Dalam kasus tersebut, Anda harus menggunakan wildcard (*).

```
"Resource": "*"
```

Untuk melihat daftar jenis sumber daya App Runner dan jenisnya ARNs, lihat Sumber daya yang ditentukan oleh AWS App Runner dalam Referensi Otorisasi Layanan. Untuk mempelajari tindakan yang dapat menentukan ARN setiap sumber daya, lihat Tindakan yang ditentukan oleh AWS App Runner.

Kunci syarat

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 `Condition` (atau blok `Condition`) akan memungkinkan Anda menentukan kondisi yang menjadi dasar suatu pernyataan berlaku. Elemen `Condition` bersifat opsional. Anda dapat membuat ekspresi bersyarat yang menggunakan operator kondisi, misalnya sama dengan atau kurang dari, untuk mencocokkan kondisi dalam kebijakan dengan nilai-nilai yang diminta.

Jika Anda menentukan beberapa elemen `Condition` dalam sebuah pernyataan, atau beberapa kunci dalam elemen `Condition` tunggal, maka AWS akan mengevaluasinya menggunakan operasi `AND` logis. Jika Anda menentukan beberapa nilai untuk satu kunci kondisi, AWS mengevaluasi kondisi menggunakan `OR` operasi logis. Semua kondisi harus dipenuhi sebelum izin pernyataan diberikan.

Anda juga dapat menggunakan variabel placeholder saat menentukan kondisi. Sebagai contoh, Anda dapat memberikan izin kepada pengguna IAM untuk mengakses sumber daya hanya jika izin tersebut mempunyai tanda yang sesuai dengan nama pengguna IAM mereka. Untuk informasi selengkapnya, lihat Elemen kebijakan IAM: variabel dan tanda dalam Panduan Pengguna IAM.

AWS mendukung kunci kondisi global dan kunci kondisi khusus layanan. Untuk melihat semua kunci kondisi AWS global, lihat kunci konteks kondisi AWS global di Panduan Pengguna IAM.

App Runner mendukung penggunaan beberapa kunci kondisi global. Untuk melihat semua kunci kondisi AWS global, lihat Kunci Konteks Kondisi AWS Global di Panduan Pengguna IAM.

App Runner mendefinisikan satu set kunci kondisi khusus layanan. Selain itu, App Runner mendukung kontrol akses berbasis tag, yang diimplementasikan menggunakan tombol kondisi. Untuk detailnya, lihat the section called “Otorisasi berdasarkan tag App Runner”.

Untuk melihat daftar kunci kondisi App Runner, lihat Kunci kondisi untuk AWS App Runner Referensi Otorisasi Layanan. Untuk mempelajari tindakan dan sumber daya yang dapat Anda gunakan kunci kondisi, lihat Tindakan yang ditentukan oleh AWS App Runner.

Contoh

Untuk melihat contoh kebijakan berbasis identitas App Runner, lihat. Contoh kebijakan berbasis identitas App Runner

Kebijakan berbasis sumber daya App Runner

App Runner tidak mendukung kebijakan berbasis sumber daya.

Otorisasi berdasarkan tag App Runner

Anda dapat melampirkan tag ke resource App Runner atau meneruskan tag dalam permintaan ke App Runner. Untuk mengendalikan akses berdasarkan tanda, berikan informasi tentang tanda di elemen kondisi dari kebijakan menggunakan kunci kondisi `apprunner:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`*, atau `aws:TagKeys`. Untuk informasi selengkapnya tentang menandai resource App Runner, lihat. the section called “Konfigurasi”

Untuk melihat contoh kebijakan berbasis identitas untuk membatasi akses ke sumber daya berdasarkan tag pada sumber daya tersebut, lihat Mengontrol akses ke layanan App Runner berdasarkan tag.

Izin pengguna App Runner

Untuk menggunakan App Runner, pengguna IAM memerlukan izin untuk tindakan App Runner. Cara umum untuk memberikan izin kepada pengguna adalah dengan melampirkan kebijakan ke pengguna atau grup IAM. Untuk informasi selengkapnya tentang mengelola izin pengguna, lihat Mengubah izin untuk pengguna IAM di Panduan Pengguna IAM.

App Runner menyediakan dua kebijakan terkelola yang dapat Anda lampirkan ke pengguna.

- `AWSAppRunnerReadOnlyAccess`— Memberikan izin untuk membuat daftar dan melihat detail tentang sumber daya Pelari Aplikasi.
- `AWSAppRunnerFullAccess`— Memberikan izin untuk semua tindakan App Runner.

Untuk kontrol izin pengguna yang lebih terperinci, Anda dapat membuat kebijakan khusus dan melampirkannya ke pengguna Anda. Untuk detailnya, lihat Membuat kebijakan IAM di Panduan Pengguna IAM.

Untuk contoh kebijakan pengguna, lihatthe section called “Kebijakan pengguna”.

AWSAppRunnerReadOnlyAccess

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "apprunner:List*",
        "apprunner:Describe*"
      ],
      "Resource": "*"
    }
  ]
}
```

AWSAppRunnerFullAccess

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:CreateServiceLinkedRole",
      "Resource": [
        "arn:aws:iam::*:role/aws-service-role/apprunner.amazonaws.com/
AWSServiceRoleForAppRunner",
        "arn:aws:iam::*:role/aws-service-role/networking.apprunner.amazonaws.com/
AWSServiceRoleForAppRunnerNetworking"
      ],
      "Condition": {
        "StringLike": {
          "iam:AWSServiceName": [
            "apprunner.amazonaws.com",
            "networking.apprunner.amazonaws.com"
          ]
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "iam:PassedToService": "apprunner.amazonaws.com"
        }
      }
    },
    {
      "Sid": "AppRunnerAdminAccess",
      "Effect": "Allow",
      "Action": "apprunner:*",
      "Resource": "*"
    }
  ]
}
```

Peran IAM Pelari Aplikasi

Peran IAM adalah entitas di dalam Anda Akun AWS yang memiliki izin khusus.

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 Runner mendukung peran terkait layanan. Untuk informasi tentang membuat atau mengelola peran terkait layanan App Runner, lihat. the section called “Menggunakan peran terkait layanan”

Peran layanan

Fitur ini memungkinkan layanan untuk menerima peran layanan atas nama Anda. Peran ini mengizinkan layanan untuk mengakses sumber daya di layanan lain untuk menyelesaikan tindakan atas nama Anda. Peran layanan muncul di akun IAM Anda dan dimiliki oleh akun tersebut. Ini berarti bahwa pengguna IAM dapat mengubah izin untuk peran ini. Namun, melakukan hal itu dapat merusak fungsionalitas layanan.

App Runner mendukung beberapa peran layanan.

Peran akses

Peran akses adalah peran yang digunakan App Runner untuk mengakses gambar di Amazon Elastic Container Registry (Amazon ECR) Registry ECR) di akun Anda. Diperlukan untuk mengakses gambar di Amazon ECR, dan tidak diperlukan dengan Amazon ECR Public. Sebelum membuat layanan berdasarkan gambar di Amazon ECR, gunakan IAM untuk membuat peran layanan dan menggunakan kebijakan `AWSAppRunnerServicePolicyForECRAccess` terkelola di dalamnya. Anda kemudian dapat meneruskan peran ini ke App Runner saat memanggil CreateServiceAPI di AuthenticationConfigurationanggota SourceConfigurationparameter, atau saat Anda menggunakan konsol App Runner untuk membuat layanan.

AWSAppRunnerServicePolicyForECRAccess

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

```
      "Action": [
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchCheckLayerAvailability",
        "ecr:BatchGetImage",
        "ecr:DescribeImages",
        "ecr:GetAuthorizationToken"
      ],
      "Resource": "*"
    }
  ]
}
```

> **Note**
>
> Jika Anda membuat kebijakan kustom sendiri untuk peran akses Anda, pastikan `"Resource": "*"` untuk menentukan `ecr:GetAuthorizationToken` tindakan tersebut. Token dapat digunakan untuk mengakses registri ECR Amazon apa pun yang dapat Anda akses.

Saat membuat peran akses, pastikan untuk menambahkan kebijakan kepercayaan yang menyatakan prinsip layanan App Runner `build.apprunner.amazonaws.com` sebagai entitas tepercaya.

Kebijakan kepercayaan untuk peran akses

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

Jika Anda menggunakan konsol App Runner untuk membuat layanan, konsol dapat secara otomatis membuat peran akses untuk Anda dan memilihnya untuk layanan baru. Konsol juga mencantumkan peran lain di akun Anda, dan Anda dapat memilih peran yang berbeda jika Anda mau.

Peran contoh

Peran instans adalah peran opsional yang digunakan App Runner untuk memberikan izin ke tindakan AWS layanan yang diperlukan instance komputasi layanan Anda. Anda perlu memberikan peran instance ke App Runner jika kode aplikasi Anda memanggil AWS actions (APIs). Sematkan izin yang diperlukan dalam peran instans Anda atau buat kebijakan kustom Anda sendiri dan gunakan dalam peran instance. Kami tidak memiliki cara untuk mengantisipasi panggilan mana yang digunakan kode Anda. Oleh karena itu, kami tidak menyediakan kebijakan terkelola untuk tujuan ini.

Sebelum membuat layanan App Runner, gunakan IAM untuk membuat peran layanan dengan kebijakan kustom atau tertanam yang diperlukan. Anda kemudian dapat meneruskan peran ini ke App Runner sebagai peran instance saat memanggil CreateServiceAPI di `InstanceRoleArn` anggota InstanceConfigurationparameter, atau saat Anda menggunakan konsol App Runner untuk membuat layanan.

Saat membuat peran instance, pastikan untuk menambahkan kebijakan kepercayaan yang menyatakan prinsip layanan App Runner `tasks.apprunner.amazonaws.com` sebagai entitas tepercaya.

Kebijakan kepercayaan untuk peran contoh

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

Jika Anda menggunakan konsol App Runner untuk membuat layanan, konsol akan mencantumkan peran di akun Anda, dan Anda dapat memilih peran yang Anda buat untuk tujuan ini.

Untuk informasi tentang membuat layanan, lihatthe section called “Pembuatan”.

Contoh kebijakan berbasis identitas App Runner

Secara default, pengguna dan peran IAM tidak memiliki izin untuk membuat atau memodifikasi AWS App Runner sumber daya. Mereka juga tidak dapat melakukan tugas menggunakan AWS Management Console, AWS CLI, atau AWS API. Administrator IAM harus membuat kebijakan IAM yang memberikan izin kepada pengguna dan peran untuk melakukan operasi API tertentu pada sumber daya yang diperlukan. Administrator kemudian harus melampirkan kebijakan tersebut ke pengguna IAM atau grup yang memerlukan izin tersebut.

Untuk mempelajari cara membuat kebijakan berbasis identitas IAM menggunakan contoh dokumen kebijakan JSON ini, lihat Membuat Kebijakan pada Tab JSON dalam Panduan Pengguna IAM.

Untuk topik keamanan App Runner lainnya, lihatKeamanan.

Topik

- Praktik terbaik kebijakan
- Kebijakan pengguna
- Mengontrol akses ke layanan App Runner berdasarkan tag

Praktik terbaik kebijakan

Kebijakan berbasis identitas menentukan apakah seseorang dapat membuat, mengakses, atau menghapus sumber daya Pelari Aplikasi 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 Anda 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 AWS CloudFormation. Untuk informasi selengkapnya, lihat Elemen kebijakan JSON IAM: Kondisi dalam Panduan Pengguna IAM.
- Gunakan IAM Access Analyzer untuk memvalidasi kebijakan IAM Anda untuk memastikan izin yang aman dan fungsional – IAM Access Analyzer memvalidasi kebijakan baru dan yang sudah ada sehingga kebijakan tersebut mematuhi bahasa kebijakan IAM (JSON) dan praktik terbaik IAM. IAM Access Analyzer menyediakan lebih dari 100 pemeriksaan kebijakan dan rekomendasi yang dapat ditindaklanjuti untuk membantu Anda membuat kebijakan yang aman dan fungsional. Untuk informasi selengkapnya, lihat Validasi kebijakan 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.

Kebijakan pengguna

Untuk mengakses konsol App Runner, pengguna IAM harus memiliki set izin minimum. Izin ini harus memungkinkan Anda untuk membuat daftar dan melihat detail tentang sumber daya Pelari Aplikasi di situs Anda. Akun AWS Jika Anda membuat kebijakan berbasis identitas yang lebih ketat daripada izin minimum yang diperlukan, konsol tidak akan berfungsi seperti yang dimaksudkan untuk pengguna dengan kebijakan tersebut.

App Runner menyediakan dua kebijakan terkelola yang dapat Anda lampirkan ke pengguna.

- `AWSAppRunnerReadOnlyAccess`— Memberikan izin untuk membuat daftar dan melihat detail tentang sumber daya Pelari Aplikasi.

- `AWSAppRunnerFullAccess`— Memberikan izin untuk semua tindakan App Runner.

Untuk memastikan bahwa pengguna dapat menggunakan konsol App Runner, lampirkan, setidaknya, kebijakan `AWSAppRunnerReadOnlyAccess` terkelola ke pengguna. Sebagai gantinya, Anda dapat melampirkan kebijakan `AWSAppRunnerFullAccess` terkelola, atau menambahkan izin tambahan tertentu, untuk memungkinkan pengguna membuat, memodifikasi, dan menghapus sumber daya. 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 gantinya, izinkan akses hanya ke tindakan yang cocok dengan operasi API yang ingin Anda izinkan untuk dilakukan pengguna.

Contoh berikut menunjukkan kebijakan pengguna kustom. Anda dapat menggunakannya sebagai titik awal untuk menentukan kebijakan pengguna kustom Anda sendiri. Salin contoh, dan atau hapus tindakan, cakup sumber daya, dan tambahkan kondisi.

Contoh: kebijakan pengguna manajemen konsol dan koneksi

Kebijakan contoh ini memungkinkan akses konsol dan memungkinkan pembuatan dan pengelolaan koneksi. Itu tidak mengizinkan pembuatan dan manajemen layanan App Runner. Hal ini dapat dilampirkan ke pengguna yang perannya adalah untuk mengelola akses layanan App Runner ke aset kode sumber.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "apprunner:List*",
        "apprunner:Describe*",
        "apprunner:CreateConnection",
        "apprunner:DeleteConnection"
      ],
      "Resource": "*"
    }
  ]
}
```

Contoh: kebijakan pengguna yang menggunakan tombol kondisi

Contoh di bagian ini menunjukkan izin bersyarat yang bergantung pada beberapa properti sumber daya atau parameter tindakan.

Kebijakan contoh ini memungkinkan pembuatan layanan App Runner tetapi menyangkal menggunakan koneksi bernama. `prod`

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCreateAppRunnerServiceWithNonProdConnections",
      "Effect": "Allow",
      "Action": "apprunner:CreateService",
      "Resource": "*",
      "Condition": {
        "ArnNotLike": {
          "apprunner:ConnectionArn": "arn:aws:apprunner:*:*:connection/prod/*"
        }
      }
    }
  ]
}
```

Kebijakan contoh ini memungkinkan memperbarui layanan App Runner yang diberi nama `preprod` hanya dengan konfigurasi penskalaan otomatis bernama. `preprod`

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowUpdatePreProdAppRunnerServiceWithPreProdASConfig",
      "Effect": "Allow",
      "Action": "apprunner:UpdateService",
      "Resource": "arn:aws:apprunner:*:*:service/preprod/*",
      "Condition": {
        "ArnLike": {
          "apprunner:AutoScalingConfigurationArn":
 "arn:aws:apprunner:*:*:autoscalingconfiguration/preprod/*"
        }
      }
```

```
    }
  ]
}
```

Mengontrol akses ke layanan App Runner berdasarkan tag

Anda dapat menggunakan kondisi dalam kebijakan berbasis identitas untuk mengontrol akses ke sumber daya App Runner berdasarkan tag. Contoh ini menunjukkan cara membuat kebijakan yang memungkinkan penghapusan layanan App Runner. Namun, izin diberikan hanya jika tanda layanan `Owner` memiliki nilai nama pengguna dari pengguna tersebut. Kebijakan ini juga memberi izin yang diperlukan untuk menyelesaikan tindakan ini pada konsol tersebut.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListServicesInConsole",
      "Effect": "Allow",
      "Action": "apprunner:ListServices",
      "Resource": "*"
    },
    {
      "Sid": "DeleteServiceIfOwner",
      "Effect": "Allow",
      "Action": "apprunner:DeleteService",
      "Resource": "arn:aws:apprunner:*:*:service/*",
      "Condition": {
        "StringEquals": {"apprunner:ResourceTag/Owner": "${aws:username}"}
      }
    }
  ]
}
```

Anda dapat melampirkan kebijakan ini ke pengguna IAM di akun Anda. Jika pengguna bernama `richard-roe` mencoba menghapus layanan App Runner, layanan harus diberi tag `Owner=richard-roe` atau. `owner=richard-roe` Jika tidak, aksesnya akan ditolak. Kunci tanda syarat `Owner` cocok dengan `Owner` dan `owner` karena nama kunci syarat tidak terpengaruh huruf besar/kecil. Untuk informasi lebih lanjut, lihat Elemen Kebijakan IAM JSON: Persyaratan dalam Panduan Pengguna IAM.

Menggunakan peran terkait layanan untuk App Runner

AWS App Runner menggunakan AWS Identity and Access Management peran terkait layanan (IAM). Peran terkait layanan adalah jenis unik peran IAM yang ditautkan langsung ke App Runner. Peran terkait layanan telah ditentukan sebelumnya oleh App Runner dan menyertakan semua izin yang diperlukan layanan untuk memanggil layanan lain AWS atas nama Anda.

Topik

- Menggunakan peran untuk manajemen
- Menggunakan peran untuk jaringan

Menggunakan peran untuk manajemen

AWS App Runner menggunakan AWS Identity and Access Management peran terkait layanan (IAM). Peran terkait layanan adalah jenis unik peran IAM yang ditautkan langsung ke App Runner. Peran terkait layanan telah ditentukan sebelumnya oleh App Runner dan menyertakan semua izin yang diperlukan layanan untuk memanggil layanan lain AWS atas nama Anda.

Peran terkait layanan membuat pengaturan App Runner lebih mudah karena Anda tidak perlu menambahkan izin yang diperlukan secara manual. App Runner mendefinisikan izin peran terkait layanan, dan kecuali ditentukan lain, hanya App Runner yang dapat mengambil perannya. Izin yang ditentukan mencakup kebijakan kepercayaan dan kebijakan izin, dan kebijakan izin tersebut tidak dapat dilampirkan ke entitas IAM lainnya.

Anda dapat menghapus peran tertaut layanan hanya setelah menghapus sumber daya terkait terlebih dahulu. Ini melindungi sumber daya Pelari Aplikasi 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 bersama tautan untuk melihat dokumentasi peran tertaut layanan untuk layanan tersebut.

Izin peran terkait layanan untuk Pelari Aplikasi

App Runner menggunakan nama peran terkait layanan. AWSServiceRoleForAppRunner

Peran ini memungkinkan App Runner untuk melakukan tugas-tugas berikut:

- Dorong log ke grup CloudWatch log Amazon Logs.
- Buat aturan CloudWatch Acara Amazon untuk berlangganan dorongan gambar Amazon Elastic Container Registry (Amazon ECR).
- Kirim informasi penelusuran ke AWS X-Ray.

Peran AWSService RoleForAppRunner terkait layanan mempercayai layanan berikut untuk mengambil peran:

- `apprunner.amazonaws.com`

Kebijakan izin peran AWSService RoleForAppRunner terkait layanan berisi semua izin yang diperlukan oleh Pelari Aplikasi untuk menyelesaikan tindakan atas nama Anda.

AppRunnerServiceRolePolicy kebijakan terkelola

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "logs:CreateLogGroup",
        "logs:PutRetentionPolicy"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:logs:*:*:log-group:/aws/apprunner/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:/aws/apprunner/*:log-stream:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "events:PutRule",
```

```
        "events:PutTargets",
        "events:DeleteRule",
        "events:RemoveTargets",
        "events:DescribeRule",
        "events:EnableRule",
        "events:DisableRule"
      ],
      "Resource": "arn:aws:events:*:*:rule/AWSAppRunnerManagedRule*"
    }
  ]
}
```

Kebijakan untuk penelusuran X-Ray

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "xray:PutTraceSegments",
        "xray:PutTelemetryRecords",
        "xray:GetSamplingRules",
        "xray:GetSamplingTargets",
        "xray:GetSamplingStatisticSummaries"
      ],
      "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 Runner

Anda tidak perlu membuat peran terkait layanan secara manual. Saat Anda membuat layanan App Runner di, API AWS Management Console, atau AWS API AWS CLI, App Runner akan membuat peran terkait layanan untuk Anda.

Jika Anda menghapus peran tertaut layanan ini, dan ingin membuatnya lagi, Anda dapat mengulangi proses yang sama untuk membuat kembali peran tersebut di akun Anda. Saat Anda membuat layanan App Runner, App Runner akan membuat peran terkait layanan untuk Anda lagi.

Mengedit peran terkait layanan untuk App Runner

App Runner tidak mengizinkan Anda mengedit peran AWSService RoleForAppRunner terkait layanan. Setelah Anda membuat peran terkait layanan, Anda tidak dapat mengubah nama peran karena berbagai entitas mungkin mereferensikan peran tersebut. Namun, Anda dapat mengedit penjelasan peran menggunakan IAM. Untuk informasi selengkapnya, lihat Mengedit Peran Tertaut Layanan dalam Panduan Pengguna IAM.

Menghapus peran terkait layanan untuk App Runner

Jika Anda tidak perlu lagi menggunakan fitur atau layanan yang memerlukan peran terkait layanan, kami merekomendasikan Anda menghapus peran tersebut. Dengan begitu, Anda tidak memiliki entitas yang tidak digunakan dan tidak dipantau atau dipelihara secara aktif. Namun, Anda harus membersihkan peran tertaut layanan terlebih dahulu sebelum dapat menghapusnya secara manual.

Membersihkan peran tertaut-layanan

Sebelum dapat menggunakan IAM untuk menghapus peran tertaut-layanan, Anda harus terlebih dahulu menghapus semua sumber daya yang digunakan oleh peran tersebut.

Di App Runner, ini berarti menghapus semua layanan App Runner di akun Anda. Untuk mempelajari cara menghapus layanan App Runner, lihat. the section called “Penghapusan”

> **Note**
>
> Jika layanan App Runner menggunakan peran saat Anda mencoba menghapus sumber daya, penghapusan mungkin gagal. Jika hal itu terjadi, tunggu beberapa menit dan coba mengoperasikannya lagi.

Untuk menghapus peran terkait layanan secara manual

Gunakan konsol IAM, the AWS CLI, atau AWS API untuk menghapus peran AWSService RoleForAppRunner terkait layanan. Untuk informasi selengkapnya, silakan lihat Menghapus Peran Terkait Layanan di Panduan Pengguna IAM.

Wilayah yang didukung untuk peran terkait layanan App Runner

App Runner mendukung penggunaan peran terkait layanan di semua wilayah tempat layanan tersedia. Untuk informasi selengkapnya, lihat AWS App Runner titik akhir dan kuota di. Referensi Umum AWS

Menggunakan peran untuk jaringan

AWS App Runner menggunakan AWS Identity and Access Management peran terkait layanan (IAM). Peran terkait layanan adalah jenis unik peran IAM yang ditautkan langsung ke App Runner. Peran terkait layanan telah ditentukan sebelumnya oleh App Runner dan menyertakan semua izin yang diperlukan layanan untuk memanggil layanan lain AWS atas nama Anda.

Peran terkait layanan membuat pengaturan App Runner lebih mudah karena Anda tidak perlu menambahkan izin yang diperlukan secara manual. App Runner mendefinisikan izin peran terkait layanan, dan kecuali ditentukan lain, hanya App Runner yang dapat mengambil perannya. Izin yang ditentukan mencakup kebijakan kepercayaan dan kebijakan izin, dan kebijakan izin tersebut tidak dapat dilampirkan ke entitas IAM lainnya.

Anda dapat menghapus peran tertaut layanan hanya setelah menghapus sumber daya terkait terlebih dahulu. Ini melindungi sumber daya Pelari Aplikasi 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 bersama tautan untuk melihat dokumentasi peran tertaut layanan untuk layanan tersebut.

Izin peran terkait layanan untuk Pelari Aplikasi

App Runner menggunakan nama peran terkait layanan. AWSServiceRoleForAppRunnerNetworking

Peran ini memungkinkan App Runner untuk melakukan tugas-tugas berikut:

- Lampirkan VPC ke layanan App Runner Anda dan kelola antarmuka jaringan.

Peran AWSService RoleForAppRunnerNetworking terkait layanan mempercayai layanan berikut untuk mengambil peran:

- `networking.apprunner.amazonaws.com`

Kebijakan izin peran bernama AppRunnerNetworkingServiceRolePolicy berisi semua izin yang dibutuhkan App Runner untuk menyelesaikan tindakan atas nama Anda.

AppRunnerNetworkingServiceRolePolicy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeVpcs",
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "ec2:CreateNetworkInterface",
      "Resource": "*",
      "Condition": {
        "ForAllValues:StringEquals": {
          "aws:TagKeys": [
            "AWSAppRunnerManaged"
          ]
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "ec2:CreateTags",
      "Resource": "arn:aws:ec2:*:*:network-interface/*",
      "Condition": {
        "StringEquals": {
          "ec2:CreateAction": "CreateNetworkInterface"
        },
        "StringLike": {
          "aws:RequestTag/AWSAppRunnerManaged": "*"
        }
      }
    },
```

```
    {
      "Effect": "Allow",
      "Action": "ec2:DeleteNetworkInterface",
      "Resource": "*",
      "Condition": {
        "Null": {
          "ec2:ResourceTag/AWSAppRunnerManaged": "false"
        }
      }
    }
  ]
}
```

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 Runner

Anda tidak perlu membuat peran terkait layanan secara manual. Saat Anda membuat konektor VPC di, API AWS Management Console, atau AWS API AWS CLI, App Runner membuat peran terkait layanan untuk Anda.

Jika Anda menghapus peran tertaut layanan ini, dan ingin membuatnya lagi, Anda dapat mengulangi proses yang sama untuk membuat kembali peran tersebut di akun Anda. Saat Anda membuat konektor VPC, App Runner membuat peran terkait layanan untuk Anda lagi.

Mengedit peran terkait layanan untuk App Runner

App Runner tidak mengizinkan Anda mengedit peran AWSService RoleForAppRunnerNetworking terkait layanan. Setelah Anda membuat peran terkait layanan, Anda tidak dapat mengubah nama peran karena berbagai entitas mungkin mereferensikan peran tersebut. Namun, Anda dapat mengedit penjelasan peran menggunakan IAM. Untuk informasi selengkapnya, lihat Mengedit Peran Tertaut Layanan dalam Panduan Pengguna IAM.

Menghapus peran terkait layanan untuk App Runner

Jika Anda tidak perlu lagi menggunakan fitur atau layanan yang memerlukan peran terkait layanan, kami merekomendasikan Anda menghapus peran tersebut. Dengan begitu, Anda tidak memiliki entitas yang tidak digunakan dan tidak dipantau atau dipelihara secara aktif. Namun, Anda harus membersihkan peran tertaut layanan terlebih dahulu sebelum dapat menghapusnya secara manual.

Membersihkan Peran Terkait Layanan

Sebelum dapat menggunakan IAM untuk menghapus peran tertaut-layanan, Anda harus terlebih dahulu menghapus semua sumber daya yang digunakan oleh peran tersebut.

Di App Runner, ini berarti melepaskan konektor VPC dari semua layanan App Runner di akun Anda, dan menghapus konektor VPC. Untuk informasi selengkapnya, lihat the section called “Lalu lintas keluar”.

> **Note**
>
> Jika layanan App Runner menggunakan peran saat Anda mencoba menghapus sumber daya, penghapusan mungkin gagal. Jika hal itu terjadi, tunggu beberapa menit dan coba mengoperasikannya lagi.

Hapus Peran Terkait Layanan secara Manual

Gunakan konsol IAM, the AWS CLI, atau AWS API untuk menghapus peran AWSService RoleForAppRunnerNetworking terkait layanan. Untuk informasi selengkapnya, silakan lihat Menghapus Peran Terkait Layanan di Panduan Pengguna IAM.

Wilayah yang didukung untuk peran terkait layanan App Runner

App Runner mendukung penggunaan peran terkait layanan di semua wilayah tempat layanan tersedia. Untuk informasi selengkapnya, lihat AWS App Runner titik akhir dan kuota di. Referensi Umum AWS

AWS kebijakan terkelola untuk AWS App Runner

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.

Pembaruan App Runner ke kebijakan AWS terkelola

Lihat detail tentang pembaruan kebijakan AWS terkelola untuk App Runner sejak layanan ini mulai melacak perubahan ini. Untuk peringatan otomatis tentang perubahan pada halaman ini, berlangganan umpan RSS di halaman riwayat Dokumen Pelari Aplikasi.

Perubahan	Deskripsi	Tanggal
AWSAppRunnerReadOnlyAccess – Kebijakan baru	App Runner menambahkan kebijakan baru untuk memungkinkan pengguna mencantumkan dan melihat detail tentang sumber daya Pelari Aplikasi.	Februari 24, 2022
AWSAppRunnerFullAccess – Pembaruan ke kebijakan yang ada	App Runner memperbarui daftar sumber daya untuk `iam:CreateServiceLinkedRole` tindakan tersebut guna memungkinkan pembuatan peran `AWSServiceRoleForAppRunnerNetworking` terkait layanan.	Februari 8, 2022
AppRunnerNetworkingServiceRolePolicy – Kebijakan baru	App Runner menambahkan kebijakan baru untuk mengizinkan App Runner melakukan panggilan ke Amazon Virtual	Februari 8, 2022

Perubahan	Deskripsi	Tanggal
	Private Cloud untuk melampirkan VPC ke layanan App Runner Anda dan mengelola antarmuka jaringan atas nama layanan App Runner. Kebijakan ini digunakan dalam peran `AWSServiceRoleForAppRunnerNetworking` terkait layanan.	
AWSAppRunnerFullAccess – Kebijakan baru	App Runner menambahkan kebijakan baru untuk memungkinkan pengguna melakukan semua tindakan App Runner.	Jan 10, 2022
AppRunnerServiceRolePolicy – Kebijakan baru	App Runner menambahkan kebijakan baru untuk mengizinkan App Runner melakukan panggilan ke Amazon CloudWatch Logs dan Amazon CloudWatch Events atas nama layanan App Runner. Kebijakan ini digunakan dalam peran `AWSServiceRoleForAppRunner` terkait layanan.	Mar 1, 2021
AWSAppRunnerServicePolicyForECRAccess – Kebijakan baru	App Runner menambahkan kebijakan baru untuk memungkinkan App Runner mengakses gambar Amazon Elastic Container Registry (Amazon ECR) Registry ECR) di akun Anda.	Mar 1, 2021
App Runner mulai melacak perubahan	App Runner mulai melacak perubahan untuk kebijakan yang AWS dikelola.	Mar 1, 2021

Memecahkan masalah identitas dan akses App Runner

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang mungkin Anda temui saat bekerja dengan AWS App Runner dan IAM.

Untuk topik keamanan App Runner lainnya, lihatKeamanan.

Topik

- Saya tidak berwenang untuk melakukan tindakan di App Runner
- Saya ingin mengizinkan orang di luar saya Akun AWS untuk mengakses sumber daya App Runner saya

Saya tidak berwenang untuk melakukan tindakan di App Runner

Jika AWS Management Console memberitahu Anda bahwa Anda tidak berwenang untuk melakukan tindakan, hubungi administrator Anda untuk bantuan. Administrator Anda adalah orang yang memberi Anda kredensi AWS masuk Anda.

Contoh error berikut terjadi ketika pengguna IAM bernama `marymajor` mencoba menggunakan konsol untuk melihat detail tentang layanan App Runner tetapi tidak memiliki `apprunner:DescribeService` izin.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 apprunner:DescribeService on resource: my-example-service
```

Dalam hal ini, Mary meminta administratornya untuk memperbarui kebijakannya untuk memungkinkannya mengakses *`my-example-service`* sumber daya menggunakan `apprunner:DescribeService` tindakan tersebut.

Saya ingin mengizinkan orang di luar saya Akun AWS untuk mengakses sumber daya App Runner 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 Runner mendukung fitur ini, lihatBagaimana App Runner 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.

Pencatatan dan pemantauan di App Runner

Pemantauan adalah bagian penting untuk menjaga keandalan, ketersediaan, dan kinerja AWS App Runner layanan Anda. Mengumpulkan data pemantauan dari semua bagian AWS solusi Anda memungkinkan Anda untuk lebih mudah men-debug kegagalan jika terjadi. App Runner terintegrasi dengan beberapa AWS alat untuk memantau layanan App Runner Anda dan menanggapi potensi insiden.

CloudWatch Alarm Amazon

Dengan CloudWatch alarm Amazon, Anda dapat menonton metrik layanan selama periode waktu yang Anda tentukan. Jika metrik melebihi ambang batas tertentu untuk jumlah periode tertentu, Anda menerima pemberitahuan.

App Runner mengumpulkan berbagai metrik tentang layanan secara keseluruhan dan instance (unit penskalaan) yang menjalankan layanan web Anda. Untuk informasi selengkapnya, lihat Metrik () CloudWatch.

Log aplikasi

App Runner mengumpulkan output dari kode aplikasi Anda dan mengalirkannya ke Amazon Logs. CloudWatch Apa yang ada dalam output ini terserah Anda. Misalnya, Anda dapat menyertakan catatan rinci permintaan yang dibuat untuk layanan web Anda. Catatan log ini mungkin terbukti berguna dalam audit keamanan dan akses. Untuk informasi selengkapnya, lihat Log (CloudWatch Log).

AWS CloudTrail log tindakan

App Runner terintegrasi dengan AWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan di App Runner. CloudTrail menangkap semua panggilan API untuk App Runner sebagai peristiwa. Anda dapat melihat peristiwa terbaru di CloudTrail konsol, dan Anda dapat membuat jejak untuk mengaktifkan pengiriman CloudTrail

acara secara berkelanjutan ke bucket Amazon Simple Storage Service (Amazon S3). Untuk informasi selengkapnya, lihat Tindakan API (CloudTrail).

Validasi kepatuhan untuk App Runner

Auditor pihak ketiga menilai keamanan dan kepatuhan AWS App Runner sebagai bagian dari beberapa program AWS kepatuhan. Program ini mencakup SOC, PCI, FedRAMP, HIPAA, dan lainnya.

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. AWS menyediakan sumber daya berikut untuk membantu kepatuhan:

- Kepatuhan dan Tata Kelola Keamanan – Panduan implementasi solusi ini membahas pertimbangan arsitektur serta memberikan langkah-langkah untuk menerapkan fitur keamanan dan kepatuhan.
- Referensi Layanan yang Memenuhi Syarat HIPAA — Daftar layanan yang memenuhi syarat HIPAA. Tidak semua memenuhi Layanan AWS syarat HIPAA.
- AWS Sumber Daya AWS — Kumpulan buku kerja dan panduan ini mungkin berlaku untuk industri dan lokasi Anda.
- AWS Panduan Kepatuhan Pelanggan - Memahami model tanggung jawab bersama melalui lensa kepatuhan. Panduan ini merangkum praktik terbaik untuk mengamankan Layanan AWS dan memetakan panduan untuk kontrol keamanan di berbagai kerangka kerja (termasuk Institut Standar dan Teknologi Nasional (NIST), Dewan Standar Keamanan Industri Kartu Pembayaran (PCI), dan Organisasi Internasional untuk Standardisasi (ISO)).
- Mengevaluasi Sumber Daya dengan Aturan dalam Panduan AWS Config Pengembang — AWS Config Layanan menilai seberapa baik konfigurasi sumber daya Anda mematuhi praktik internal, pedoman industri, dan peraturan.
- AWS Security Hub— Ini Layanan AWS memberikan pandangan komprehensif tentang keadaan keamanan Anda di dalamnya AWS. Security Hub menggunakan kontrol keamanan untuk sumber

daya AWS Anda serta untuk memeriksa kepatuhan Anda terhadap standar industri keamanan dan praktik terbaik. Untuk daftar layanan dan kontrol yang didukung, lihat Referensi kontrol Security Hub.

- Amazon GuardDuty — Ini Layanan AWS mendeteksi potensi ancaman terhadap beban kerja Akun AWS, kontainer, dan data Anda dengan memantau lingkungan Anda untuk aktivitas mencurigakan dan berbahaya. GuardDuty dapat membantu Anda mengatasi berbagai persyaratan kepatuhan, seperti PCI DSS, dengan memenuhi persyaratan deteksi intrusi yang diamanatkan oleh kerangka kerja kepatuhan tertentu.
- AWS Audit ManagerIni Layanan AWS membantu Anda terus mengaudit AWS penggunaan Anda untuk menyederhanakan cara Anda mengelola risiko dan kepatuhan terhadap peraturan dan standar industri.

Untuk topik keamanan App Runner lainnya, lihatKeamanan.

Ketahanan di App Runner

Infrastruktur AWS global dibangun di sekitar Wilayah AWS dan Availability Zones. Wilayah AWS menyediakan beberapa Availability Zone yang terpisah secara fisik dan terisolasi, yang terhubung dengan latensi rendah, throughput tinggi, dan jaringan yang sangat redundan. Dengan Zona Ketersediaan, Anda dapat merancang dan mengoperasikan aplikasi dan basis data yang secara otomatis melakukan failover di antara Zona Ketersediaan tanpa gangguan. Zona Ketersediaan memiliki ketersediaan dan toleransi kesalahan yang lebih baik, dan dapat diskalakan dibandingkan infrastruktur biasa yang terdiri dari satu atau beberapa pusat data.

Untuk informasi selengkapnya tentang Wilayah AWS dan Availability Zone, lihat Infrastruktur AWS Global.

AWS App Runner mengelola dan mengotomatiskan penggunaan infrastruktur AWS global atas nama Anda. Saat menggunakan App Runner, Anda mendapat manfaat dari ketersediaan dan mekanisme toleransi kesalahan yang AWS ditawarkan.

Untuk topik keamanan App Runner lainnya, lihatKeamanan.

Keamanan infrastruktur di AWS App Runner

Sebagai layanan terkelola, AWS App Runner dilindungi oleh prosedur keamanan jaringan AWS global yang dijelaskan dalam whitepaper Amazon Web Services: Tinjauan Proses Keamanan.

Anda menggunakan panggilan API yang AWS dipublikasikan untuk mengakses App Runner melalui jaringan. Klien harus mendukung Keamanan Lapisan Pengangkutan (TLS) 1.2 atau versi yang lebih baru. Klien juga harus mendukung cipher suite dengan perfect forward secrecy (PFS) seperti Ephemeral Diffie-Hellman (DHE) atau Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Sebagian besar sistem modern seperti Java 7 dan sistem yang lebih baru mendukung mode ini.

Selain itu, permintaan harus ditandatangani menggunakan ID kunci akses dan kunci akses rahasia yang terkait dengan prinsipal IAM. Atau Anda dapat menggunakan AWS Security Token Service (AWS STS) untuk menghasilkan kredensial keamanan sementara untuk menandatangani permintaan.

Untuk topik keamanan App Runner lainnya, lihatKeamanan.

Menggunakan App Runner dengan titik akhir VPC

AWS Aplikasi Anda mungkin mengintegrasikan AWS App Runner layanan dengan layanan lain Layanan AWS yang berjalan di VPC dari Amazon Virtual Private Cloud (Amazon VPC). Bagian dari aplikasi Anda mungkin membuat permintaan ke App Runner dari dalam VPC. Misalnya, Anda dapat menggunakan AWS CodePipeline untuk terus menerapkan ke layanan App Runner Anda. Salah satu cara untuk meningkatkan keamanan aplikasi Anda adalah dengan mengirim permintaan App Runner ini (dan permintaan ke yang lain Layanan AWS) melalui titik akhir VPC.

Menggunakan titik akhir VPC, Anda dapat menghubungkan VPC Anda secara pribadi ke layanan endpoint VPC yang didukung Layanan AWS dan VPC yang didukung oleh. AWS PrivateLink Anda tidak memerlukan gateway internet, perangkat NAT, koneksi VPN, atau AWS Direct Connect koneksi.

Sumber daya di VPC Anda tidak menggunakan alamat IP publik untuk berinteraksi dengan sumber daya App Runner. Lalu lintas antara VPC dan App Runner tidak meninggalkan jaringan Amazon. Untuk informasi selengkapnya tentang titik akhir VPC, lihat titik akhir VPC di Panduan.AWS PrivateLink

> **Note**
>
> Secara default, aplikasi web di layanan App Runner berjalan di VPC yang disediakan dan dikonfigurasi oleh App Runner. VPC ini bersifat publik. Ini berarti bahwa itu terhubung ke internet. Anda dapat secara opsional mengaitkan aplikasi Anda dengan VPC khusus. Untuk informasi selengkapnya, lihat the section called “Lalu lintas keluar”.
> Anda dapat mengonfigurasi layanan Anda untuk mengakses internet, termasuk AWS APIs, bahkan ketika layanan Anda terhubung ke VPC. Untuk petunjuk tentang cara mengaktifkan

akses internet publik untuk lalu lintas keluar VPC, lihat. the section called “Pertimbangan saat memilih subnet ”
App Runner tidak mendukung pembuatan titik akhir VPC untuk aplikasi Anda.

Menyiapkan titik akhir VPC untuk App Runner

Untuk membuat titik akhir VPC antarmuka untuk layanan App Runner di VPC Anda, ikuti prosedur Buat titik akhir antarmuka di Panduan.AWS PrivateLink Untuk Nama Layanan, pilih`com.amazonaws.`*`region`*`.apprunner`.

Pertimbangan privasi jaringan VPC

> **Important**
>
> Menggunakan titik akhir VPC untuk App Runner tidak memastikan bahwa semua lalu lintas dari VPC Anda tetap berada di luar internet. VPC mungkin bersifat publik. Selain itu, beberapa bagian dari solusi Anda mungkin tidak menggunakan titik akhir VPC untuk melakukan AWS panggilan API. Misalnya, Layanan AWS mungkin memanggil layanan lain menggunakan titik akhir publik mereka. Jika privasi lalu lintas diperlukan untuk solusi di VPC Anda, baca bagian ini.

Untuk memastikan privasi lalu lintas jaringan di VPC Anda, pertimbangkan hal berikut:

- Aktifkan nama DNS — Bagian dari aplikasi Anda mungkin masih mengirim permintaan ke App Runner melalui internet menggunakan titik akhir `apprunner.`*`region`*`.amazonaws.com` publik. Jika VPC Anda dikonfigurasi dengan akses internet, permintaan ini berhasil tanpa indikasi kepada Anda. Anda dapat mencegah hal ini dengan memastikan bahwa Aktifkan nama DNS diaktifkan saat Anda membuat titik akhir. Secara default, ini disetel ke true. Ini menambahkan entri DNS di VPC Anda yang memetakan titik akhir layanan publik untuk antarmuka VPC endpoint.
- Konfigurasikan titik akhir VPC untuk layanan tambahan — Solusi Anda mungkin mengirim permintaan ke yang lain. Layanan AWS Misalnya, AWS CodePipeline mungkin mengirim permintaan ke AWS CodeBuild. Konfigurasikan titik akhir VPC untuk layanan ini, dan aktifkan nama DNS pada titik akhir ini.
- Konfigurasikan VPC pribadi — Jika memungkinkan (jika solusi Anda tidak memerlukan akses internet sama sekali), atur VPC Anda sebagai pribadi, yang berarti tidak memiliki koneksi internet.

Ini memastikan bahwa titik akhir VPC yang hilang menyebabkan kesalahan yang terlihat, sehingga Anda dapat menambahkan titik akhir yang hilang.

Menggunakan kebijakan titik akhir untuk mengontrol akses dengan VPC endpoint

Kebijakan titik akhir VPC didukung untuk App Runner. Secara default, akses penuh ke App Runner diizinkan melalui titik akhir antarmuka. Kebijakan titik akhir VPC dapat digunakan untuk mengontrol prinsipal AWS mana yang dapat mengakses titik akhir App Runner. Atau, Anda dapat mengaitkan grup keamanan dengan antarmuka jaringan titik akhir untuk mengontrol lalu lintas ke App Runner melalui titik akhir antarmuka.

Mengintegrasikan dengan titik akhir antarmuka

Dukungan App Runner AWS PrivateLink, yang menyediakan konektivitas pribadi ke App Runner dan menghilangkan paparan lalu lintas ke internet. Untuk mengaktifkan aplikasi Anda mengirim permintaan ke App Runner menggunakan AWS PrivateLink, konfigurasikan jenis titik akhir VPC yang dikenal sebagai titik akhir antarmuka. Untuk informasi selengkapnya, lihat Titik akhir VPC Antarmuka (AWS PrivateLink) di Panduan.AWS PrivateLink

Analisis konfigurasi dan kerentanan di App Runner

AWS dan pelanggan kami berbagi tanggung jawab untuk mencapai tingkat keamanan dan kepatuhan komponen perangkat lunak yang tinggi. Untuk informasi lebih lanjut, lihat model tanggung jawab AWS bersama.

Gambar wadah tambalan

Menambal gambar kontainer adalah bagian dari tanggung jawab pelanggan dalam model keamanan bersama. Pemilik gambar bertanggung jawab untuk memperbarui dan secara teratur menambal gambar kontainer. Sebaiknya buat jadwal rutin untuk memeriksa dan menerapkan pembaruan pada gambar kontainer Anda. Untuk informasi selengkapnya tentang cara memindai gambar Anda dari kerentanan, lihat AWS App Runner Documentation

Untuk topik keamanan App Runner lainnya, lihatKeamanan.

Praktik terbaik keamanan untuk App Runner

AWS App Runner menyediakan beberapa fitur keamanan untuk dipertimbangkan saat Anda mengembangkan dan menerapkan kebijakan keamanan Anda sendiri. Praktik terbaik berikut adalah pedoman umum dan tidak mewakili solusi keamanan yang lengkap. Karena praktik terbaik ini mungkin tidak sesuai atau cukup untuk lingkungan Anda, anggap sebagai pertimbangan yang membantu, dan bukan sebagai resep.

Untuk topik keamanan App Runner lainnya, lihatKeamanan.

Praktik terbaik keamanan pencegahan

Kontrol keamanan preventif berusaha mencegah insiden sebelum terjadi.

Terapkan akses hak akses paling rendah

App Runner menyediakan kebijakan terkelola AWS Identity and Access Management (IAM) untuk pengguna IAM dan peran akses. Kebijakan terkelola ini menentukan semua izin yang mungkin diperlukan untuk pengoperasian layanan App Runner Anda dengan benar.

Aplikasi Anda mungkin tidak memerlukan semua izin dalam kebijakan terkelola kami. Anda dapat menyesuaikannya dan hanya memberikan izin yang diperlukan bagi pengguna dan layanan App Runner untuk menjalankan tugasnya. Hal ini sangat relevan untuk kebijakan pengguna, di mana peran pengguna yang berbeda mungkin memiliki kebutuhan izin yang berbeda. Menerapkan akses hak akses paling rendah adalah hal mendasar dalam mengurangi risiko keamanan dan dampak yang dapat diakibatkan oleh kesalahan atau niat jahat.

Pindai gambar Anda untuk kerentanan

Anda dapat menggunakan Amazon ECR APIs untuk membantu mengidentifikasi kerentanan perangkat lunak dalam gambar kontainer Anda. Untuk informasi selengkapnya, lihat dokumentasi Amazon ECR.

Praktik terbaik keamanan detective

Kontrol keamanan detective mengidentifikasi pelanggaran keamanan setelah mereka telah terjadi. Mereka dapat membantu Anda mendeteksi potensi ancaman keamanan atau insiden.

Melaksanakan pemantauan

Pemantauan merupakan bagian penting dalam menjaga keandalan, keamanan, ketersediaan, dan kinerja solusi App Runner Anda. AWS menyediakan beberapa alat dan layanan untuk membantu Anda memantau AWS layanan Anda.

Berikut adalah beberapa instans item yang perlu dipantau:

- CloudWatch Metrik Amazon untuk App Runner — Setel alarm untuk metrik utama App Runner dan untuk metrik kustom aplikasi Anda. Untuk detailnya, lihat Metrik () CloudWatch.
- AWS CloudTrail entri — Melacak tindakan yang mungkin memengaruhi ketersediaan, seperti `PauseService` atau`DeleteConnection`. Untuk detailnya, lihat Tindakan API (CloudTrail).

AWS Glosarium

Untuk AWS terminologi terbaru, lihat AWS glosarium di Referensi.Glosarium AWS

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