Panduan Pengguna

AWS Proton

AWS Proton: Panduan Pengguna

Table of Contents

..... ix
Apa itu AWS Proton? 1
Tim platform 1
Developer 2
Alur kerja 2
Panduan Pengakhiran dan Migrasi 3
Status Layanan Hingga Deprecation 3
Informasi Migrasi Penting 4
Solusi Alternatif 4
Panduan Migrasi 6
FAQs 7
Menyiapkan 8
Menyiapkan dengan IAM 8
Mendaftar untuk AWS 9
Mmebuat pengguna IAM 9
Peran layanan 10
Menyiapkan dengan AWS Proton 11
Menyiapkan bucket Amazon S3 12
Menyiapkan AWS CodeStar koneksi 12
Menyiapkan pengaturan CI/CD saluran pipa akun 13
Menyiapkan AWS CLI 15
Memulai 16
Prasyarat 16
Memulai alur kerja 17
Memulai dengan konsol 18
Langkah 1: Buka AWS Proton konsol 19
Langkah 2: Bersiaplah untuk menggunakan contoh templat 19
Langkah 3: Buat template lingkungan 19
Langkah 4: Buat template layanan 20
Langkah 5: Ciptakan lingkungan 21
Langkah 6: Opsional - Buat layanan dan gunakan aplikasi 22
Langkah 7: Bersihkan. 24
Memulai dengan CLI 25
1. Daftarkan templat lingkungan 25

2. Daftarkan templat layanan ... 26
3. Menyebarkan lingkungan ... 27
4. Menyebarkan layanan ... 28
5. Bersihkan ... 30
Pustaka template ... 31
Bagaimana cara AWS Proton kerja ... 33
Objek ... 34
Metode penyediaan ... 37
AWS-penyediaan terkelola ... 39
CodeBuild penyediaan ... 41
Penyediaan yang dikelola sendiri ... 44
AWS Proton terminologi ... 47
Penulisan templat dan bundel ... 49
Bundel template ... 49
Parameter ... 51
Jenis parameter ... 51
Menggunakan parameter ... 52
Parameter lingkungan CloudFormation IAc ... 56
Parameter layanan CloudFormation iAc ... 61
Parameter komponen CloudFormation IAc ... 63
CloudFormation filter parameter ... 67
CodeBuild parameter penyediaan ... 74
Parameter Terraform IAc ... 76
Infrastruktur sebagai file kode ... 77
CloudFormation File iAc ... 78
CodeBuild bundel ... 131
File Terraform IAc ... 137
Berkas skema ... 145
Persyaratan skema lingkungan ... 145
Persyaratan skema layanan ... 149
Manifestasikan dan bungkus ... 152
Paket template lingkungan membungkus ... 155
Paket template layanan membungkus ... 156
Pertimbangan bundel template ... 156
Template ... 158
Versi ... 159

Publikasikan 161
Publikasikan templat lingkungan 161
Publikasikan templat layanan 168
Lihat template 177
Perbarui template 181
Hapus template 183
Konfigurasi sinkronisasi templat 187
Mendorong komit 187
Menyinkronkan templat layanan 188
Pertimbangan sinkronisasi template 188
Buat 189
Tampilan 196
Edit 197
Delete 198
Konfigurasi sinkronisasi layanan 199
AWS Proton Berkas OPS 199
Buat 203
Tayang 205
Sunting 206
Delete 207
Lingkungan 209
Peran IAM 209
AWS Proton peran layanan 209
Buat 210
Membuat dan menyediakan di akun yang sama 212
Buat di satu akun dan ketentuan di akun lain 214
Penyediaan yang dikelola sendiri 219
Tampilan 222
Perbarui 223
Memperbarui lingkungan penyediaan AWS terkelola 225
Memperbarui lingkungan penyediaan yang dikelola sendiri 227
Membatalkan penerapan lingkungan yang sedang berlangsung 231
Delete 233
Koneksi akun 235
Buat lingkungan dengan koneksi akun lingkungan 237
Kelola koneksi akun lingkungan 238

Dikelola pelanggan 244
Menggunakan lingkungan yang dikelola pelanggan 245
CodeBuild pembuatan peran penyediaan 247
Layanan 251
Buat 251
Apa yang ada dalam layanan? 252
Templat layanan 252
Membuat layanan 253
Tayang 257
Sunting 259
Edit deskripsi layanan 259
Menambah atau menghapus instance layanan 261
Delete 268
Lihat contoh 269
Perbarui contoh 271
Perbarui pipa 277
Komponen-komponen 285
Komponen vs. sumber daya lainnya 287
AWS Proton konsol 288
AWS Proton API dan AWS CLI 289
Komponen FAQ 290
Status komponen 291
File komponen iAc 293
Menggunakan parameter dengan komponen 293
Menulis file IAc yang kuat 293
CloudFormation Contoh komponen 294
Langkah-langkah administrator 295
Langkah-langkah pengembang 298
Repositori 301
Buat tautan repositori 302
Lihat data repositori tertaut 304
Hapus tautan repositori 307
Memantau 309
Otomatisasi dengan AWS Proton EventBridge 309
Tipe peristiwa 309
AWS Proton contoh acara 312

EventBridgeTutorial: Kirim peringatan Amazon Simple Notification Service untuk AWS Proton perubahan status layanan 314
Prasyarat 314
Langkah 1: Buat dan berlangganan ke topik Amazon SNS 314
Langkah 2: Mendaftarkan aturan peristiwa 315
Langkah 3: Uji aturan acara Anda 316
Langkah 4: Membersihkan 317
AWS Proton dasbor 318
AWS Proton konsol 318
Keamanan 321
Identity and Access Management 322
Audiens 322
Mengautentikasi dengan identitas 322
Mengelola akses menggunakan kebijakan 324
Bagaimana AWS Proton bekerja dengan IAM 326
Contoh kebijakan 331
AWS kebijakan terkelola 346
Menggunakan Peran Terkait Layanan 357
Pemecahan masalah 361
Konfigurasi dan analisis kerentanan 363
Perlindungan data 364
Enkripsi sisi server saat istirahat 365
Enkripsi bergerak 365
AWS Proton manajemen kunci enkripsi 365
AWS Proton konteks enkripsi 365
Keamanan infrastruktur 367
Titik akhir VPC (AWS PrivateLink) 367
Pencatatan log dan pemantauan 370
Ketahanan 371
AWS Proton cadangan 371
Praktik terbaik keamanan 371
Gunakan IAM untuk mengontrol akses 372
Jangan menanamkan kredensi di template dan bundel template Anda 372
Gunakan enkripsi untuk melindungi data sensitif 372
Gunakan AWS CloudTrail untuk melihat dan mencatat panggilan API 373
Pencegahan "confused deputy" lintas layanan 373

Codebuild dukungan kustom 374
Memperbarui Template Lingkungan 375
Penandaan 379
AWS penandaan 379
AWS Proton penandaan 380
AWS Proton AWS tag terkelola 380
Perbanyakan tag ke sumber daya yang disediakan 381
Tag terkelola pelanggan 384
Buat tag menggunakan konsol dan CLI 384
Buat tag menggunakan AWS Proton AWS CLI 386
Pemecahan Masalah 387
Kesalahan penerapan yang mereferensikan parameter CloudFormation dinamis 387
AWS Proton kuota 389
Riwayat dokumen 390
AWS Glosarium 395

Pemberitahuan akhir dukungan: Pada 7 Oktober 2026, AWS akan mengakhiri dukungan untuk AWS Proton. Setelah 7 Oktober 2026, Anda tidak akan lagi dapat mengakses AWS Proton konsol atau AWS Proton sumber daya. Infrastruktur yang Anda gunakan akan tetap utuh. Untuk informasi selengkapnya, lihat Panduan AWS Proton Pengakhiran Layanan dan Migrasi.

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

Apa itu AWS Proton?

AWS Proton adalah:

- Infrastruktur otomatis sebagai penyediaan kode dan penyebaran aplikasi tanpa server dan berbasis kontainer

  AWS Proton Layanan ini adalah kerangka kerja otomatisasi dua cabang. Sebagai administrator, Anda membuat templat layanan berversi yang menentukan infrastruktur standar dan perkakas penerapan untuk aplikasi tanpa server dan berbasis container. Sebagai pengembang aplikasi, Anda dapat memilih dari templat layanan yang tersedia untuk mengotomatiskan penerapan aplikasi atau layanan Anda.

  AWS Proton mengidentifikasi semua instance layanan yang ada yang menggunakan versi template yang sudah ketinggalan zaman untuk Anda. Sebagai administrator, Anda dapat meminta AWS Proton untuk memutakhirkannya dengan satu klik.
- Infrastruktur standar

  Tim platform dapat menggunakan AWS Proton dan memversikan infrastruktur sebagai templat kode. Mereka dapat menggunakan template ini untuk mendefinisikan dan mengelola tumpukan aplikasi standar yang berisi arsitektur, sumber daya infrastruktur, dan pipeline penyebaran CI/CD perangkat lunak.
- Penerapan terintegrasi dengan CI/CD

  Saat pengembang menggunakan antarmuka AWS Proton swalayan untuk memilih template layanan, mereka memilih definisi tumpukan aplikasi standar untuk penerapan kode mereka. AWS Proton secara otomatis menyediakan sumber daya, mengkonfigurasi CI/CD pipeline, dan menyebarkan kode ke dalam infrastruktur yang ditentukan.

AWS Proton untuk tim platform

Sebagai administrator, Anda atau anggota tim platform Anda, buat templat lingkungan dan templat layanan yang berisi infrastruktur sebagai kode. Template lingkungan mendefinisikan infrastruktur bersama yang digunakan oleh beberapa aplikasi atau sumber daya. Template layanan mendefinisikan jenis infrastruktur yang diperlukan untuk menyebarkan dan memelihara satu aplikasi atau layanan mikro dalam suatu lingkungan. AWS Proton Layanan adalah instantiasi template layanan, yang biasanya mencakup beberapa instance layanan dan pipeline. Instance AWS Proton

layanan adalah instantiasi template layanan di lingkungan tertentu. Anda atau orang lain dalam tim Anda dapat menentukan template lingkungan mana yang kompatibel dengan template layanan yang diberikan. Untuk informasi selengkapnya tentang template, lihatAWS Proton template.

Anda dapat menggunakan infrastruktur berikut sebagai penyedia kode dengan AWS Proton:

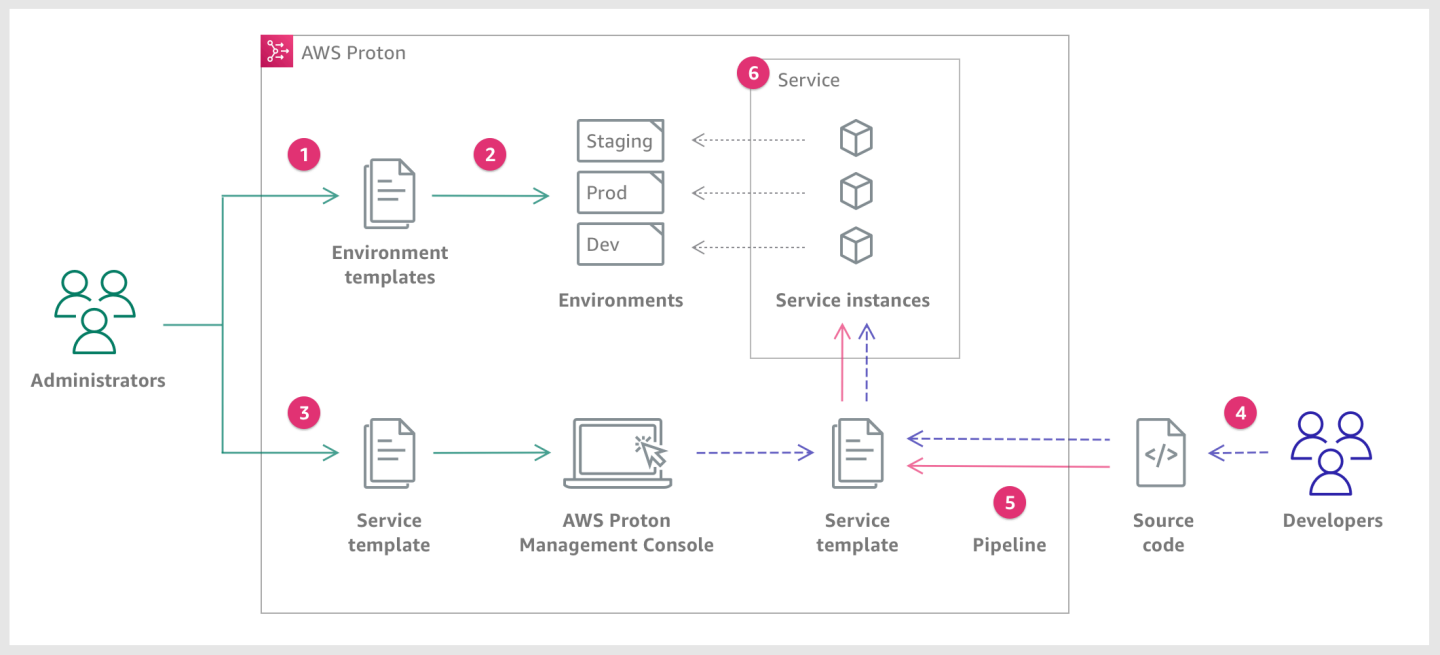
- CloudFormation
- Terraform

AWS Proton untuk pengembang

Sebagai pengembang aplikasi, Anda memilih template layanan standar yang AWS Proton digunakan untuk membuat layanan yang menyebarkan dan mengelola aplikasi Anda dalam instance layanan. AWS Proton Layanan adalah instantiasi template layanan, yang biasanya mencakup beberapa instance layanan dan pipeline.

AWS Proton alur kerja

Diagram berikut adalah visualisasi dari AWS Proton konsep-konsep utama yang dibahas dalam paragraf sebelumnya. Ini juga menawarkan gambaran tingkat tinggi tentang apa yang merupakan alur kerja sederhana AWS Proton .



1. Sebaga

Administrator, Anda membuat dan mendaftarkan Template Lingkungan dengan AWS Proton, yang mendefinisikan sumber daya bersama.

2. AWS

Proton menyebarkan satu atau beberapa Lingkungan, berdasarkan Template Lingkungan.

3. Sebaga

Administrator, Anda membuat dan mendaftarkan Template Layanan AWS Proton, yang mendefinisikan infrastruktur, pemantauan, dan CI/CD sumber daya terkait serta Template Lingkungan yang kompatibel.

4. Sebaga

Pengembang, Anda memilih Template Layanan terdaftar dan memberikan tautan ke repositori kode Sumber Anda.

5. AWS Proton menyediakan Layanan dengan Pipa CI/CD untuk instans Layanan Anda.

6. AWS Proton menyediakan dan mengelola Layanan dan Instans Layanan yang menjalankan kode Sumber seperti yang didefinisikan dalam Templat Layanan yang dipilih. Instance Layanan adalah instantiasi dari Template Layanan yang dipilih di Lingkungan untuk satu tahap Pipeline (misalnya Prod).

AWS Proton Panduan Pengakhiran Layanan dan Migrasi

AWS telah memutuskan untuk berhenti AWS Proton, dengan dukungan berakhir pada 7 Oktober 2026. Pelanggan baru tidak akan dapat mendaftar setelah 7 Oktober 2025, tetapi pelanggan yang sudah ada dapat terus menggunakan layanan ini hingga 7 Oktober 2026.

Status Layanan Hingga Deprecation

Hingga 7 Oktober 2026, AWS Proton pelanggan yang sudah ada dapat terus menggunakan layanan ini secara normal. Selama periode ini, AWS akan:

1. Menyediakan patch keamanan dan perbaikan bug kritis
2. Menjaga ketersediaan dan kinerja layanan
3. Terus menawarkan dukungan melalui AWS Dukungan saluran
4. Tidak menambahkan fitur baru ke layanan

Informasi Migrasi Penting

AWS Proton terutama CI/CD alat untuk menyebarkan infrastruktur. Ketika AWS Proton tidak digunakan lagi, CloudFormation tumpukan yang Anda gunakan dan sumber daya yang mereka kelola akan tetap utuh dan terus berfungsi. Pengakhiran hanya memengaruhi saluran pipa pengiriman dan AWS Proton layanan itu sendiri, bukan infrastruktur yang Anda gunakan.

Solusi Alternatif

Kami telah mengidentifikasi beberapa alternatif AWS Proton yang dapat membantu Anda mempertahankan infrastruktur Anda sebagai kode dan kemampuan CI/CD.

CloudFormation Sinkronisasi Git

Terbaik untuk: Tim CloudFormation yang menggunakan siapa yang menginginkan GitOps alur kerja

Sinkronisasi Git memungkinkan tim platform untuk memodelkan CloudFormation template dalam repositori git yang dapat di-fork oleh tim pengembangan. Pengembang memperbarui file parameter, mendorong perubahan ke repositori bercabang mereka, dan sinkronisasi Git memperbarui tumpukan.

Manfaat Utama:

1. Pengalaman pengembang serupa dengan AWS Proton
2. Memanfaatkan pengetahuan yang ada CloudFormation
3. Pemisahan yang jelas antara platform dan tim pengembang

Pembatasan:

1. Tidak ada konsep lingkungan
2. Tidak ada fitur pipa lanjutan
3. Bergantung pada GitHub fitur yang mungkin tidak tersedia di penyedia Git lainnya

Pelajari lebih lanjut: Sinkronisasi Git

Harmonix Aktif AWS

Terbaik untuk: Perusahaan yang membutuhkan portal pengembang internal yang komprehensif

Harmonix adalah AWS Partner solusi berdasarkan BackStage.io dan menyediakan AWS plugin yang memungkinkan tim membuat template, lingkungan, dan layanan yang mirip dengan Proton.

Manfaat Utama:

1. Fungsionalitas serupa dengan AWS Proton
2. Dibangun di atas kerangka Backstage populer
3. Pengalaman portal pengembang lengkap

Pembatasan:

1. Tidak dikelola oleh Layanan AWS tim
2. Implementasi referensi yang mungkin memerlukan kustomisasi

Pelajari lebih lanjut: https://harmonixonaws.io/

AWS CodePipeline dan AWS CodeBuild

Terbaik untuk: Tim yang membutuhkan fleksibilitas dan kontrol maksimum

Gunakan CI/CD layanan AWS dasar untuk mereplikasi AWS Proton fungsionalitas dengan fleksibilitas dan kontrol yang lebih besar.

Manfaat Utama:

1. Fleksibilitas maksimum
2. Integrasi mendalam dengan AWS layanan
3. Pemeliharaan aktif dan fitur baru

Pembatasan:

1. Membutuhkan lebih banyak pekerjaan implementasi

2. Lebih sedikit out-of-box layanan mandiri pengembang

Pelajari lebih lanjut:

Apa itu AWS CodePipeline

Apa itu AWS CodeBuild

GitHub Tindakan

Terbaik untuk: Tim yang lebih kecil menggunakan GitHub siapa yang menginginkan kesederhanaan

>Manfaat Utama

1. Integrasi mudah dengan GitHub repositori
2. Pengaturan sederhana untuk GitHub pengguna
3. Pasar besar tindakan yang dapat digunakan kembali

Pembatasan:

1. Terikat dengan GitHub ekosistem
2. Mungkin memerlukan lebih banyak pekerjaan untuk kontrol tim platform

Pelajari lebih lanjut:

GitHub Dokumentasi tindakan

Contoh CI/CD: Mengintegrasikan dengan GitHub Tindakan - CI/CD pipeline untuk menyebarkan Aplikasi Web ke Amazon EC2

Panduan Migrasi

Proses migrasi tergantung pada implementasi Anda dan alternatif yang dipilih. Langkah-langkah umum:

1. Inventarisasi sumber daya Proton Anda:
2. Pilih solusi alternatif:
3. Ekstrak data template Anda:

4. Terapkan alternatif yang Anda pilih:
5. Migrasikan beban kerja produksi:

Untuk bantuan migrasi tertentu, hubungi AWS Dukungan atau tim akun Anda.

FAQs

T: Mengapa AWS berhenti? AWS Proton J: Kami telah mengidentifikasi peluang yang lebih baik untuk memenuhi kebutuhan pelanggan akan Infrastruktur sebagai penegakan kebijakan Kode melalui AWS Partner solusi AWS dan solusi lain.

T: Apakah infrastruktur saya yang ada akan terus berfungsi setelah tanggal penghentian? A: Ya. AWS Proton Pada dasarnya adalah CI/CD alat. CloudFormation Tumpukan yang Anda gunakan dan sumber daya yang mereka kelola akan tetap utuh dan terus berfungsi. Pengakhiran hanya memengaruhi saluran pipa pengiriman, bukan infrastruktur yang Anda gunakan.

T: Bagaimana saya bisa mendapatkan bantuan terkait migrasi? A: AWS Dukungan dapat membantu migrasi Anda. Silakan hubungi AWS Dukungan, atau Anda dapat menghubungi Akun AWS manajer Anda untuk meminta bantuan.

T: Alternatif mana yang harus saya pilih? J: Alternatif terbaik tergantung pada kasus penggunaan spesifik Anda:

1. Untuk GitOps alur kerja sederhana: CloudFormation Git Sync
2. Untuk perusahaan yang membutuhkan portal pengembang: Harmonix On AWS
3. Untuk fleksibilitas maksimum: AWS CodePipeline dan AWS CodeBuild
4. Untuk tim yang sudah aktif GitHub: GitHub Tindakan

T: Apa yang terjadi jika saya tidak bermigrasi sebelum 7 Oktober 2026? A: Anda tidak lagi dapat mengakses AWS Proton. Infrastruktur Anda yang ada akan terus berfungsi, tetapi Anda tidak akan dapat menggunakannya AWS Proton untuk mengelola atau memperbaruinya.

T: Berapa lama data saya akan disimpan? A: Hingga 7 Oktober 2026. Setelah tanggal ini, semua data akan dihapus.

Jika Anda memiliki pertanyaan tambahan, silakan hubungi AWS Dukungan.

Menyiapkan

Selesaikan tugas di bagian ini sehingga Anda dapat membuat dan mendaftarkan templat layanan dan lingkungan. Anda memerlukan ini untuk menyebarkan lingkungan dan layanan dengan AWS Proton.

> **Note**
>
> Kami menawarkan tanpa AWS Proton biaya tambahan. Anda dapat membuat, mendaftar, dan memelihara templat layanan dan lingkungan tanpa biaya. Anda juga dapat mengandalkan AWS Proton untuk mengelola sendiri operasinya sendiri, seperti penyimpanan, keamanan, dan penyebaran. Satu-satunya biaya yang Anda keluarkan saat menggunakan AWS Proton adalah sebagai berikut.
>
> - Biaya penyebaran dan penggunaan AWS Cloud sumber daya yang Anda instruksikan AWS Proton untuk menyebarkan dan memelihara untuk Anda.
> - Biaya pemeliharaan AWS CodeStar koneksi ke repositori kode Anda.
> - Biaya pemeliharaan bucket Amazon S3, jika Anda menggunakan bucket untuk memberikan input. AWS Proton Anda dapat menghindari biaya ini jika Anda beralih the section called “Konfigurasi sinkronisasi templat” menggunakan repositori Git untuk Anda. the section called “Bundel template”

Topik

- Menyiapkan dengan IAM
- Menyiapkan dengan AWS Proton

Menyiapkan dengan IAM

Ketika Anda mendaftar AWS, Anda Akun AWS secara otomatis mendaftar untuk semua layanan di AWS, termasuk AWS Proton. Anda hanya dikenakan biaya untuk layanan dan sumber daya yang Anda gunakan.

> **Note**
>
> Anda dan tim Anda, termasuk administrator dan pengembang, semuanya harus berada di bawah akun yang sama.

Mendaftar untuk 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.

Mmebuat pengguna IAM

Untuk membuat pengguna administrator, pilih salah satu opsi berikut.

Pilih salah satu cara untuk mengelola administrator Anda	Untuk	Oleh	Anda juga bisa
Di Pusat Identitas IAM (Direkomendasikan)	Gunakan kredensi jangka pendek untuk mengakses. AWS Ini sejalan dengan praktik terbaik keamanan. Untuk informasi tentang praktik terbaik, lihat Praktik terbaik keamanan di IAM di Panduan Pengguna IAM.	Mengikuti petunjuk di Memulai di Panduan AWS IAM Identity Center Pengguna.	Konfigurasikan akses terprogram dengan Mengonfigurasi AWS CLI yang akan digunakan AWS IAM Identity Center dalam AWS Command Line Interface Panduan Pengguna.
Di IAM (Tidak direkomendasikan)	Gunakan kredensi jangka panjang untuk mengakses. AWS	Mengikuti petunjuk di Buat pengguna IAM untuk akses darurat di Panduan Pengguna IAM.	Konfigurasikan akses terprogram dengan Mengelola kunci akses untuk pengguna IAM di Panduan Pengguna IAM.

Menyiapkan peran AWS Proton layanan

Ada beberapa peran IAM yang mungkin ingin Anda buat untuk berbagai bagian AWS Proton solusi Anda. Anda dapat membuatnya terlebih dahulu menggunakan konsol IAM, atau Anda dapat menggunakan AWS Proton konsol untuk membuatnya untuk Anda.

Buat peran AWS Proton lingkungan untuk memungkinkan AWS Proton Anda melakukan panggilan API ke layanan komputasi dan penyimpanan lainnya Layanan AWS CloudFormation AWS CodeBuild, seperti, dan berbagai layanan komputasi dan penyimpanan, atas nama Anda untuk menyediakan sumber daya untuk Anda. Peran penyediaan yang AWS dikelola diperlukan saat lingkungan atau instance layanan apa pun yang berjalan di dalamnya menggunakan -managed provisioning.AWS CodeBuildPeran diperlukan ketika lingkungan atau salah satu contoh layanannya menggunakan CodeBuildpenyediaan. Untuk mempelajari lebih lanjut tentang peran AWS Proton lingkungan, lihatthe section called “Peran IAM”. Saat membuat lingkungan, Anda dapat menggunakan AWS Proton konsol untuk memilih peran yang ada untuk salah satu dari dua peran ini, atau untuk membuat peran dengan hak administratif untuk Anda.

Demikian pula, buat peran AWS Proton pipeline AWS Proton untuk memungkinkan melakukan panggilan API ke layanan lain atas nama Anda untuk menyediakan pipeline CI/CD untuk Anda. Untuk mempelajari lebih lanjut tentang peran AWS Proton pipeline, lihatthe section called “Peran layanan pipa”. Untuk informasi selengkapnya tentang mengonfigurasi CI/CD setelan, lihatthe section called “Menyiapkan pengaturan CI/CD saluran pipa akun”.

> **Note**
>
> Karena kami tidak tahu sumber daya mana yang akan Anda tentukan di AWS Proton template, peran yang Anda buat menggunakan konsol memiliki izin luas dan dapat digunakan sebagai peran layanan AWS Proton pipeline dan peran AWS Proton layanan. Untuk penerapan produksi, sebaiknya Anda memasukkan izin ke sumber daya spesifik yang akan diterapkan dengan membuat kebijakan khusus untuk peran layanan AWS Proton pipeline dan peran layanan lingkungan. AWS Proton Anda dapat membuat dan menyesuaikan peran ini dengan menggunakan AWS CLI atau IAM. Untuk informasi selengkapnya, lihat Peran layanan untuk AWS Proton dan Membuat layanan.

Menyiapkan dengan AWS Proton

Jika Anda ingin menggunakan AWS CLI to run AWS Proton APIs, verifikasi bahwa Anda telah menginstalnya. Jika Anda belum menginstalnya, lihatMenyiapkan AWS CLI.

AWS Proton konfigurasi spesifik:

- Untuk membuat dan mengelola template:
  - Jika Anda menggunakan konfigurasi sinkronisasi templat, siapkan AWS CodeStar koneksi.

  - Jika tidak, siapkan bucket Amazon S3.
- Untuk penyediaan infrastruktur:
  - Untuk penyediaan yang dikelola sendiri, Anda harus mengatur koneksi.AWS CodeStar
- (Opsional) Untuk menyediakan jaringan pipa:
  - Untuk penyediaan AWS-managed dan CodeBuildbased provisioning, siapkan peran pipeline.
  - Untuk penyediaan yang dikelola sendiri, siapkan repositori pipeline.

Untuk informasi selengkapnya tentang metode penyediaan, lihat. the section called “AWS-penyediaan terkelola”

Menyiapkan bucket Amazon S3

Untuk menyiapkan bucket S3, ikuti petunjuk di Buat bucket S3 pertama Anda untuk menyiapkan bucket S3. Tempatkan input Anda ke AWS Proton dalam ember di mana AWS Proton dapat mengambilnya. Masukan ini dikenal sebagai bundel template. Anda dapat mempelajari lebih lanjut tentang mereka di bagian lain dari panduan ini.

Menyiapkan AWS CodeStar koneksi

Untuk menyambung AWS Proton ke repositori, Anda membuat AWS CodeStar koneksi yang mengaktifkan pipeline saat komit baru dibuat pada repositori kode sumber pihak ketiga.

AWS Proton menggunakan koneksi ke:

- Aktifkan pipeline layanan saat komit baru dibuat pada kode sumber repositori Anda.
- Buat permintaan tarik pada infrastruktur sebagai repositori kode.
- Buat template baru versi minor atau mayor setiap kali komit didorong ke repositori template yang mengubah salah satu template Anda, jika versi tersebut belum ada.

Anda dapat terhubung ke repositori Bitbucket GitHub, GitHub Enterprise dan GitHub Enterprise Server dengan. CodeConnections Untuk informasi selengkapnya, lihat CodeConnections di AWS CodePipeline Panduan Pengguna.

Untuk mengatur CodeStar koneksi.

1. Buka konsol AWS Proton.

2. Di panel navigasi, pilih Pengaturan dan kemudian Koneksi repositori untuk membawa Anda ke halaman Koneksi di Pengaturan Alat Pengembang. Halaman ini menampilkan daftar koneksi.
3. Pilih Buat koneksi dan ikuti petunjuknya.

Menyiapkan pengaturan CI/CD saluran pipa akun

AWS Proton dapat menyediakan CI/CD pipeline untuk menyebarkan kode aplikasi ke instance layanan Anda. AWS Proton Pengaturan yang Anda perlukan untuk penyediaan pipeline bergantung pada metode penyediaan yang Anda pilih untuk pipeline Anda.

AWS-penyediaan terkelola dan CodeBuild berdasar—mengatur peran pipa

Dengan penyediaan dan CodeBuild penyediaan yang AWS dikelola, menyediakan saluran pipa untuk Anda. AWS Proton Oleh karena itu, AWS Proton diperlukan peran layanan yang memberikan izin untuk penyediaan saluran pipa. Masing-masing dari dua metode penyediaan ini menggunakan peran layanannya sendiri. Peran ini dibagikan di semua pipeline AWS Proton layanan dan Anda mengonfigurasinya sekali di pengaturan akun Anda.

Untuk membuat peran layanan pipeline menggunakan konsol

1. Buka konsol AWS Proton.
2. Di panel navigasi, pilih Pengaturan, lalu pilih Pengaturan akun.
3. Di halaman CI/CD Pengaturan akun, pilih Konfigurasi.
4. Lakukan salah satu tindakan berikut:
   - Untuk AWS Proton membuat peran layanan pipeline untuk Anda

     [Untuk mengaktifkan penyediaan saluran pipa yang AWS dikelola] Di halaman Konfigurasi setelan akun, di bagian peran pipeline penyediaan yang AWS dikelola:

     a. Pilih Peran layanan baru.
     b. Masukkan nama untuk peran tersebut, misalnya,**`myProtonPipelineServiceRole`**.
     c. Centang kotak centang untuk menyetujui membuat AWS Proton peran dengan hak administratif di akun Anda.

     [Untuk mengaktifkan penyediaan saluran pipa CodeBuild berbasis] Di halaman Konfigurasi setelan akun, di bagian peran CodeBuild pipeline, pilih Peran layanan yang ada, dan pilih

peran layanan yang Anda buat di bagian peran CloudFormation pipeline. Atau, jika Anda tidak menetapkan peran CloudFormation pipeline, ulangi tiga langkah sebelumnya untuk membuat peran layanan baru.

- Untuk memilih peran layanan pipeline yang ada

  [Untuk mengaktifkan penyediaan saluran pipa yang AWS dikelola] Di halaman Konfigurasi setelan akun, di bagian peran pipeline penyediaan AWS-terkelola, pilih Peran layanan yang ada, dan pilih peran layanan di akun Anda. AWS

  [Untuk mengaktifkan CodeBuild penyediaan pipeline] Di halaman Konfigurasi setelan akun, di bagian peran penyediaan CodeBuild pipeline, pilih Peran layanan yang ada, dan pilih peran layanan di akun Anda. AWS

5. Pilih Simpan perubahan.

   Peran layanan pipeline baru Anda ditampilkan di halaman Pengaturan akun.

Penyediaan yang dikelola sendiri—menyiapkan repositori pipa

Dengan penyediaan yang dikelola sendiri, AWS Proton mengirimkan permintaan tarik (PR) ke repositori penyediaan yang telah Anda siapkan, dan kode otomatisasi Anda bertanggung jawab untuk menyediakan saluran pipa. Oleh karena itu, AWS Proton tidak memerlukan peran layanan untuk menyediakan jaringan pipa. Sebaliknya, ia membutuhkan repositori penyediaan terdaftar. Kode otomatisasi Anda di repositori harus mengambil peran yang sesuai yang memberikan izin untuk menyediakan saluran pipa.

Untuk mendaftarkan repositori penyediaan pipeline menggunakan konsol

1. Buat repositori penyediaan CI/CD pipeline jika Anda belum membuatnya. Untuk informasi selengkapnya tentang pipeline dalam penyediaan yang dikelola sendiri, lihat. the section called “Penyediaan yang dikelola sendiri”
2. Di panel navigasi, pilih Pengaturan, lalu pilih Pengaturan akun.
3. Di halaman CI/CD Pengaturan akun, pilih Konfigurasi.
4. Di halaman Konfigurasi pengaturan akun, di bagian repositori pipa CI/CD:

   a. Pilih Repositori baru, lalu pilih salah satu penyedia repositori.

   b. Untuk CodeStar koneksi, pilih salah satu koneksi Anda.

> **Note**
>
> Jika Anda belum memiliki koneksi ke akun penyedia repositori yang relevan, pilih Tambahkan CodeStar koneksi baru, selesaikan proses pembuatan koneksi, lalu pilih tombol refresh di sebelah menu CodeStarkoneksi. Anda sekarang harus dapat memilih koneksi baru Anda di menu.

c. Untuk nama Repositori, pilih repositori penyediaan pipeline Anda. Menu drop-down menunjukkan daftar repositori di akun penyedia.

d. Untuk nama Branch, pilih salah satu cabang repositori.

5. Pilih Simpan perubahan.

   Repositori pipeline Anda ditampilkan di halaman Pengaturan akun.

Menyiapkan AWS CLI

Untuk menggunakan panggilan AWS CLI untuk melakukan AWS Proton API, verifikasi bahwa Anda telah menginstal versi terbaru dari AWS CLI. Untuk informasi selengkapnya, lihat Memulai AWS CLI dalam Panduan Pengguna AWS Command Line Interface . Kemudian, untuk mulai menggunakan AWS CLI with AWS Proton, lihatthe section called “Memulai dengan CLI”.

Memulai dengan AWS Proton

Sebelum memulai, siapkan untuk menggunakan AWS Proton dan verifikasi bahwa Anda telah memenuhi prasyarat Memulai.

Mulailah AWS Proton dengan memilih satu atau beberapa jalur berikut:

- Ikuti konsol contoh terpandu atau alur kerja CLI melalui tautan dokumentasi.
- Jalankan melalui alur kerja konsol contoh yang dipandu.
- Jalankan melalui AWS CLI alur kerja contoh yang dipandu.

Topik

- Prasyarat
- Memulai alur kerja
- Memulai dengan Konsol Manajemen AWS
- Memulai dengan AWS CLI
- Pustaka AWS Proton template

Prasyarat

Sebelum Anda mulai menggunakan AWS Proton, pastikan bahwa prasyarat berikut terpenuhi. Untuk informasi selengkapnya, lihat Menyiapkan.

- Anda memiliki akun IAM dengan izin administrator. Untuk informasi selengkapnya, lihat Menyiapkan dengan IAM.
- Anda memiliki peran AWS Proton layanan dan peran layanan AWS Proton pipeline dilampirkan ke akun Anda. Untuk informasi selengkapnya, lihat Menyiapkan peran AWS Proton layanan dan Peran layanan untuk AWS Proton.
- Anda memiliki AWS CodeStar koneksi. Untuk informasi selengkapnya, lihat Menyiapkan AWS CodeStar koneksi.
- Anda sudah familiar dengan membuat CloudFormation template dan parameterisasi Jinja. Untuk informasi lebih lanjut, lihat Apa itu CloudFormation? di Panduan CloudFormation Pengguna dan situs web Jinja.

- Anda memiliki pengetahuan tentang layanan AWS infrastruktur.
- Anda masuk ke Akun AWS.

Memulai alur kerja

Pelajari cara membuat bundel templat, membuat dan mendaftarkan templat, dan membuat lingkungan dan layanan dengan mengikuti contoh langkah dan tautan.

Sebelum memulai, verifikasi bahwa Anda membuat peran AWS Proton layanan.

Jika templat layanan menyertakan pipeline AWS Proton layanan, verifikasi bahwa Anda telah membuat AWS CodeStar koneksi dan peran layanan AWS Proton pipeline.

Untuk informasi selengkapnya, lihat Referensi API AWS Proton layanan.

Contoh: Memulai alur kerja

1. Lihat diagram Bagaimana cara AWS Proton kerja untuk tampilan AWS Proton input dan output tingkat tinggi.
2. Buat bundel lingkungan dan bundel template layanan.
    a. Identifikasi parameter input.
    b. Buat file skema.
    c. Buat infrastruktur sebagai file kode (IAc).
    d. Untuk membungkus bundel template Anda, buat file manifes dan atur file IAC Anda, file manifes, dan file skema dalam direktori.
    e. Buat bundel template Anda dapat diakses AWS Proton.
3. Buat dan daftarkan versi template lingkungan dengan AWS Proton.

    Saat Anda menggunakan konsol untuk membuat dan mendaftarkan templat, versi templat dibuat secara otomatis.

    Bila Anda menggunakan AWS CLI untuk membuat dan mendaftarkan template:

    a. Buat template lingkungan.
    b. Buat versi template lingkungan.

Untuk informasi selengkapnya, lihat CreateEnvironmentTemplatedan CreateEnvironmentTemplateVersiondi referensi AWS Proton API.

4. Publikasikan template lingkungan Anda untuk membuatnya tersedia untuk digunakan.

   Untuk informasi selengkapnya, lihat UpdateEnvironmentTemplateVersiondi referensi AWS Proton API.

5. Untuk membuat lingkungan, pilih versi template lingkungan yang dipublikasikan dan berikan nilai untuk input yang diperlukan.

   Untuk informasi selengkapnya, lihat CreateEnvironmentdi referensi AWS Proton API.

6. Buat dan daftarkan versi template layanan dengan AWS Proton.

   Saat Anda menggunakan konsol untuk membuat dan mendaftarkan templat, versi templat dibuat secara otomatis.

   Bila Anda menggunakan AWS CLI untuk membuat dan mendaftarkan template:

   a. Buat template layanan.
   b. Buat versi template layanan.

   Untuk informasi selengkapnya, lihat CreateServiceTemplatedan CreateServiceTemplateVersiondi referensi AWS Proton API.

7. Publikasikan template layanan Anda agar tersedia untuk digunakan.

   Untuk informasi selengkapnya, lihat UpdateServiceTemplateVersiondi referensi AWS Proton API.

8. Untuk membuat layanan, pilih versi template layanan yang diterbitkan dan berikan nilai untuk input yang diperlukan.

   Untuk informasi selengkapnya, lihat CreateServicedi referensi AWS Proton API.

Memulai dengan Konsol Manajemen AWS

Memulai dengan AWS Proton

- Buat dan lihat template lingkungan.

- Buat, lihat, dan publikasikan template layanan yang menggunakan template lingkungan yang baru saja Anda buat.
- Buat lingkungan dan layanan (opsional).
- Hapus template layanan, template lingkungan, lingkungan dan layanan, jika dibuat.

Langkah 1: Buka AWS Proton konsol

- Buka konsol AWS Proton

Langkah 2: Bersiaplah untuk menggunakan contoh templat

1. Buat Koneksi Codestar ke Github dan beri nama koneksi. my-proton-connection
2. Arahkan ke https://github.com/aws-samples/aws-proton-cloudformation-sample-templates
3. Buat fork repositori di akun Github Anda.

Langkah 3: Buat template lingkungan

Di panel navigasi, pilih Template lingkungan.

1. Di halaman Template Lingkungan, pilih Buat template Lingkungan.
2. Di halaman Buat templat lingkungan, di bagian Opsi templat, pilih Buat templat untuk menyediakan lingkungan baru.
3. Di bagian sumber bundel Template, pilih Sinkronkan bundel template dari Git.
4. Di bagian repositori definisi Template, pilih Pilih repositori Git yang ditautkan.
5. Pilih my-proton-connectiondari daftar Repositori.
6. Pilih main dari daftar Branch.
7. Di bagian detail templat lingkungan Proton.
    a. Masukkan nama template sebagai**`fargate-env`**.
    b. Masukkan nama tampilan template lingkungan sebagai**`My Fargate Environment`**.
    c. (Opsional) Masukkan deskripsi untuk template lingkungan.
8. (Opsional) Di bagian Tag, pilih Tambahkan tag baru dan masukkan kunci dan nilai untuk membuat tag terkelola pelanggan.

9. Pilih template Create Environment.

   Anda sekarang berada di halaman baru yang menampilkan status dan detail untuk template lingkungan baru Anda. Rincian ini mencakup daftar AWS dan tag yang dikelola pelanggan. AWS Proton secara otomatis menghasilkan tag AWS terkelola untuk Anda saat Anda membuat AWS Proton sumber daya. Untuk informasi selengkapnya, lihat AWS Proton sumber daya dan penandaan.
10. Status status template lingkungan baru dimulai di status Draft. Anda dan orang lain dengan `proton:CreateEnvironment` izin dapat melihat dan mengaksesnya. Ikuti langkah selanjutnya untuk membuat template tersedia untuk orang lain.
11. Di bagian Versi Template, pilih tombol radio di sebelah kiri versi minor template yang baru saja Anda buat (1.0). Sebagai alternatif, Anda dapat memilih Publikasikan di spanduk peringatan info dan lewati langkah berikutnya.
12. Di bagian Versi templat, pilih Publikasikan.
13. Status template berubah menjadi Diterbitkan. Karena ini adalah versi terbaru dari template, itu adalah versi Rekomendasi.
14. Di panel navigasi, pilih Template lingkungan.

    Halaman baru menampilkan daftar template lingkungan Anda bersama dengan detail template.

Langkah 4: Buat template layanan

Buat template layanan.

1. Di panel navigasi, pilih Templat layanan.
2. Di halaman Templat layanan, pilih Buat template Layanan.
3. Di halaman Buat template layanan, di bagian sumber bundel Template, pilih Sinkronkan bundel template dari Git.
4. Di bagian Template, pilih Pilih repositori Git yang ditautkan.
5. Pilih my-proton-connectiondari daftar Repositori.
6. Pilih main dari daftar Branch.
7. Di bagian detail templat layanan Proton.

   a. Masukkan nama template layanan sebagai**`backend-fargate-svc`**.

   b. Masukkan nama tampilan template layanan sebagai**`My Fargate Service`**.

c. (Opsional) Masukkan deskripsi untuk template layanan.

8. Di bagian Template lingkungan yang kompatibel.

   - Centang kotak centang di sebelah kiri templat lingkungan Lingkungan Fargate Saya untuk memilih templat lingkungan yang kompatibel untuk templat layanan baru.

9. Untuk pengaturan Enkripsi, pertahankan defaultnya.
10. Di bagian definisi Pipeline.

    - Jaga agar template ini menyertakan tombol CI/CD pipeline yang dipilih.

11. Pilih Buat template layanan.

    Anda sekarang berada di halaman baru yang menampilkan status dan detail untuk template layanan baru Anda, termasuk daftar AWS dan tag yang dikelola pelanggan.

12. Status status template layanan baru dimulai dalam status Draf. Hanya administrator yang dapat melihat dan mengaksesnya. Untuk membuat template layanan tersedia untuk digunakan oleh pengembang, ikuti langkah berikutnya.
13. Di bagian Versi Template, pilih tombol radio di sebelah kiri versi minor template yang baru saja Anda buat (1.0). Sebagai alternatif, Anda dapat memilih Publikasikan di spanduk peringatan info dan lewati langkah berikutnya.
14. Di bagian Versi templat, pilih Publikasikan.
15. Status template berubah menjadi Diterbitkan.

    Versi minor pertama dari template layanan Anda diterbitkan dan tersedia untuk digunakan oleh pengembang. Karena ini adalah versi terbaru dari template, itu adalah versi Rekomendasi.

16. Di panel navigasi, pilih Templat layanan.

    Halaman baru menampilkan daftar template dan detail layanan Anda.

Langkah 5: Ciptakan lingkungan

Pada panel navigasi, pilih Lingkungan.

1. Pilih Buat lingkungan.
2. Di halaman Pilih template lingkungan, pilih template yang baru saja Anda buat. Ini bernama My Fargate Environment. Kemudian, pilih Konfigurasi.

3. Di halaman Configure environment, di bagian Provisioning, pilih Provisioning through. AWS Proton
4. Di bagian Akun Deployment, pilih Ini Akun AWS.
5. Di Pengaturan Lingkungan, masukkan nama lingkungan sebagai**`my-fargate-environment`**.
6. Di bagian Peran lingkungan, pilih Peran layanan baru atau, jika Anda telah membuat peran AWS Proton layanan, pilih Peran layanan yang ada.

   a. Pilih Peran layanan baru untuk membuat peran baru.

      i. Masukkan nama peran Lingkungan sebagai**`MyProtonServiceRole`**.

      ii. Centang kotak centang untuk menyetujui membuat peran AWS Proton layanan dengan hak administratif untuk akun Anda.

   b. Pilih Peran layanan yang ada untuk menggunakan peran yang ada.

      • Pilih peran Anda di bidang tarik-turun nama peran Lingkungan.
7. Pilih Berikutnya.
8. Pada halaman Konfigurasi pengaturan kustom, gunakan default.
9. Pilih Berikutnya dan tinjau masukan Anda.
10. Pilih Buat.

    Lihat detail dan status lingkungan, serta tag AWS terkelola dan tag terkelola pelanggan untuk lingkungan Anda.
11. Pada panel navigasi, pilih Lingkungan.

    Halaman baru menampilkan daftar lingkungan Anda bersama dengan status dan detail lingkungan lainnya.

Langkah 6: Opsional - Buat layanan dan gunakan aplikasi

1. Buka konsol AWS Proton.
2. Pada panel navigasi, silakan pilih Layanan.
3. Di halaman Layanan, pilih Buat layanan.
4. Di halaman Pilih templat layanan, pilih template Layanan Fargate Saya dengan memilih tombol radio di sudut kanan atas kartu templat.
5. Pilih Konfigurasi di sudut kanan bawah halaman.

6. Di halaman Konfigurasi Layanan, di bagian Pengaturan layanan, masukkan nama layanan**`my-service`**.
7. (Opsional) Masukkan deskripsi untuk layanan.
8. Di bagian Pengaturan Repositori Layanan:
   a. Untuk CodeStar koneksi, pilih koneksi Anda dari daftar.
   b. Untuk nama Repositori, pilih nama repositori kode sumber Anda dari daftar.
   c. Untuk nama Branch, pilih nama cabang repositori kode sumber Anda dari daftar.
9. (Opsional) Di bagian Tag, pilih Tambahkan tag baru dan masukkan kunci dan nilai untuk membuat tag terkelola pelanggan. Lalu pilih Selanjutnya.
10. Di halaman Konfigurasi pengaturan kustom, di bagian Instans layanan, di bagian Instans baru, ikuti langkah selanjutnya untuk memberikan nilai khusus untuk parameter instance layanan Anda.
    a. Masukkan nama instance**`my-app-service`**.
    b. Pilih lingkungan **`my-fargate-environment`** untuk instance layanan Anda.
    c. Simpan default untuk parameter instance yang tersisa.
    d. Simpan default untuk input Pipeline.
    e. Pilih Berikutnya dan tinjau masukan Anda.
    f. Pilih Buat dan lihat status dan detail layanan Anda.
11. Di halaman detail layanan, lihat status instans dan pipeline layanan Anda dengan memilih tab Ikhtisar dan Pipeline. Pada halaman ini Anda juga dapat melihat AWS dan tag yang dikelola pelanggan. AWS Proton secara otomatis membuat tag AWS terkelola untuk Anda. Pilih Kelola tag untuk membuat dan memodifikasi tag terkelola pelanggan. Untuk informasi lebih lanjut tentang penandaan, lihat AWS Proton sumber daya dan penandaan.
12. Setelah layanan Aktif, di tab Ikhtisar, di bagian Instans Layanan, pilih nama instance layanan Anda. my-app-service

    Anda sekarang berada di halaman detail instance layanan.
13. Untuk melihat aplikasi Anda, di bagian Output, salin ServiceEndpointtautan ke browser Anda.

    Anda melihat AWS Proton grafik di halaman web.
14. Setelah layanan dibuat, di panel navigasi, pilih Layanan untuk melihat daftar layanan Anda.

Langkah 7: Bersihkan.

1. Buka konsol AWS Proton.
2. Hapus layanan (jika Anda membuatnya)
   a. Pada panel navigasi, silakan pilih Layanan.
   b. Di halaman Layanan, pilih nama layanan my-service.

      Anda sekarang berada di halaman detail layanan untuk layanan saya.
   c. Di sudut kanan atas halaman, pilih Tindakan dan kemudian Hapus.
   d. Modal meminta Anda untuk mengonfirmasi tindakan hapus.
   e. Ikuti instruksi dan pilih Ya, hapus.
3. Hapus lingkungan
   a. Pada panel navigasi, pilih Lingkungan.
   b. Di halaman Lingkungan, pilih tombol radio di sebelah kiri lingkungan yang baru saja Anda buat.
   c. Pilih Actions (Tindakan), lalu Delete (Hapus).
   d. Modal meminta Anda untuk mengonfirmasi tindakan hapus.
   e. Ikuti instruksi dan pilih Ya, hapus.
4. Hapus templat layanan
   a. Di panel navigasi, pilih Templat layanan.
   b. Di halaman Templat layanan, pilih tombol radio di sebelah kiri templat layanan my-svc-template.
   c. Pilih Actions (Tindakan), lalu Delete (Hapus).
   d. Modal meminta Anda untuk mengonfirmasi tindakan hapus.
   e. Ikuti instruksi dan pilih Ya, hapus. Ini menghapus template layanan dan semua versinya.
5. Hapus templat lingkungan
   a. Di panel navigasi, pilih Template lingkungan.
   b. Di halaman Template Lingkungan, pilih tombol radio di sebelah kiri my-env-template.
   c. Pilih Actions (Tindakan), lalu Delete (Hapus).
   d. Modal meminta Anda untuk mengonfirmasi tindakan hapus.

e. Ikuti instruksi dan pilih Ya, hapus. Ini menghapus template lingkungan dan semua versinya.

6. Hapus Koneksi Codestar Anda

Memulai dengan AWS CLI

Untuk memulai dengan AWS Proton menggunakan AWS CLI, ikuti tutorial ini. Tutorial ini menunjukkan AWS Proton layanan load-balanced yang dihadapi publik berdasarkan. AWS Fargate Tutorial ini juga menyediakan CI/CD pipeline yang menyebarkan situs web statis dengan gambar yang ditampilkan.

Sebelum Anda mulai, pastikan Anda sudah diatur dengan benar. Lihat perinciannya di the section called “Prasyarat”.

Langkah 1: Daftarkan template lingkungan

Pada langkah ini, sebagai administrator, Anda mendaftarkan contoh template lingkungan, yang berisi cluster Amazon Elastic Container Service (Amazon ECS) Container Service (Amazon ECS) dan Amazon Virtual Private Cloud (Amazon VPC) dengan dua subnet. public/private

Untuk mendaftarkan template lingkungan

1. Garpu repositori CloudFormation Template AWS Proton Sample ke GitHub akun atau organisasi Anda. Repositori ini mencakup template lingkungan dan layanan yang kita gunakan dalam tutorial ini.

   Kemudian, daftarkan repositori bercabang Anda dengan. AWS Proton Untuk informasi selengkapnya, lihat the section called “Buat tautan repositori”.

2. Buat template lingkungan.

   Sumber daya template lingkungan melacak versi template lingkungan.

   ```
   $ aws proton create-environment-template \
     --name "fargate-env" \
     --display-name "Public VPC Fargate" \
     --description "VPC with public access and ECS cluster"
   ```

3. Buat konfigurasi sinkronisasi templat.

AWS Proton mengatur hubungan sinkronisasi antara repositori dan template lingkungan Anda. Kemudian membuat template versi 1.0 dalam DRAFT status.

```
$ aws proton create-template-sync-config \
  --template-name "fargate-env" \
  --template-type "ENVIRONMENT" \
  --repository-name "your-forked-repo" \
  --repository-provider "GITHUB" \
  --branch "your-branch" \
  --subdirectory "environment-templates/fargate-env"
```

4. Tunggu hingga versi template lingkungan berhasil didaftarkan.

   Ketika perintah ini kembali dengan status keluar dari`0`, pendaftaran versi selesai. Ini berguna dalam skrip untuk memastikan Anda berhasil menjalankan perintah di langkah berikutnya.

```
$ aws proton wait environment-template-version-registered \
  --template-name "fargate-env" \
  --major-version "1" \
  --minor-version "0"
```

5. Publikasikan versi template lingkungan agar tersedia untuk pembuatan lingkungan.

```
$ aws proton update-environment-template-version \
  --template-name "fargate-env" \
  --major-version "1" \
  --minor-version "0" \
  --status "PUBLISHED"
```

Langkah 2: Daftarkan template layanan

Pada langkah ini, sebagai administrator, Anda mendaftarkan contoh templat layanan, yang berisi semua sumber daya yang diperlukan untuk menyediakan layanan Amazon ECS Fargate di belakang penyeimbang beban dan pipeline CI/CD yang digunakan. AWS CodePipeline

Untuk mendaftarkan template layanan

1. Buat template layanan.

   Sumber daya template layanan melacak versi template layanan.

```
$ aws proton create-service-template \
  --name "load-balanced-fargate-svc" \
  --display-name "Load balanced Fargate service" \
  --description "Fargate service with an application load balancer"
```

2. Buat konfigurasi sinkronisasi templat.

   AWS Proton mengatur hubungan sinkronisasi antara repositori dan template layanan Anda. Kemudian membuat template versi 1.0 dalam DRAFT status.

```
$ aws proton create-template-sync-config \
  --template-name "load-balanced-fargate-svc" \
  --template-type "SERVICE" \
  --repository-name "your-forked-repo" \
  --repository-provider "GITHUB" \
  --branch "your-branch" \
  --subdirectory "service-templates/load-balanced-fargate-svc"
```

3. Tunggu hingga versi template layanan berhasil didaftarkan.

   Ketika perintah ini kembali dengan status keluar dari`0`, pendaftaran versi selesai. Ini berguna dalam skrip untuk memastikan Anda berhasil menjalankan perintah di langkah berikutnya.

```
$ aws proton wait service-template-version-registered \
  --template-name "load-balanced-fargate-svc" \
  --major-version "1" \
  --minor-version "0"
```

4. Publikasikan versi template layanan untuk membuatnya tersedia untuk pembuatan layanan.

```
$ aws proton update-service-template-version \
  --template-name "load-balanced-fargate-svc" \
  --major-version "1" \
  --minor-version "0" \
  --status "PUBLISHED"
```

Langkah 3: Menyebarkan lingkungan

Pada langkah ini, sebagai administrator, Anda membuat instance AWS Proton lingkungan dari template lingkungan.

Untuk menyebarkan lingkungan

1. Dapatkan contoh file spesifikasi untuk template lingkungan yang Anda daftarkan.

   Anda dapat mengunduh file `environment-templates/fargate-env/spec/spec.yaml` dari repositori contoh template. Atau, Anda dapat mengambil seluruh repositori secara lokal dan menjalankan create-environment perintah dari direktori. `environment-templates/fargate-env`

2. Buat lingkungan.

   AWS Proton membaca nilai input dari spesifikasi lingkungan Anda, menggabungkannya dengan template lingkungan Anda, dan menyediakan sumber daya lingkungan di AWS akun Anda menggunakan peran AWS Proton layanan Anda.

   ```
   $ aws proton create-environment \
     --name "fargate-env-prod" \
     --template-name "fargate-env" \
     --template-major-version 1 \
     --proton-service-role-arn "arn:aws:iam::123456789012:role/AWS ProtonServiceRole"
    \
     --spec "file://spec/spec.yaml"
   ```

3. Tunggu hingga lingkungan berhasil diterapkan.

   ```
   $ aws proton wait environment-deployed --name "fargate-env-prod"
   ```

Langkah 4: Menyebarkan layanan [pengembang aplikasi]

Pada langkah sebelumnya, administrator mendaftarkan dan menerbitkan template layanan dan menerapkan lingkungan. Sebagai pengembang aplikasi, Anda sekarang dapat membuat AWS Proton layanan dan menyebarkannya ke lingkungan AWS Proton

Untuk menyebarkan layanan

1. Dapatkan contoh file spesifikasi untuk template layanan yang didaftarkan administrator.

   Anda dapat mengunduh file `service-templates/load-balanced-fargate-svc/spec/spec.yaml` dari repositori contoh template. Atau, Anda dapat mengambil seluruh repositori

secara lokal dan menjalankan create-service perintah dari direktori. `service-templates/load-balanced-fargate-svc`

2. Memindahkan repositori Layanan AWS Proton Sampel ke GitHub akun atau organisasi Anda. Repositori ini mencakup kode sumber aplikasi yang kita gunakan dalam tutorial ini.
3. Buat sebuah layanan.

   AWS Proton membaca nilai input dari spesifikasi layanan Anda, menggabungkannya dengan templat layanan Anda, dan menyediakan sumber daya layanan di AWS akun Anda di lingkungan yang ditentukan dalam spesifikasi. AWS CodePipeline Pipeline menyebarkan kode aplikasi Anda dari repositori yang Anda tentukan dalam perintah.

   ```
   $ aws proton create-service \
     --name "static-website" \
     --repository-connection-arn \
       "arn:aws:codestar-connections:us-east-1:123456789012:connection/your-codestar-connection-id" \
     --repository-id "your-GitHub-account/aws-proton-sample-services" \
     --branch-name "main" \
     --template-major-version 1 \
     --template-name "load-balanced-fargate-svc" \
     --spec "file://spec/spec.yaml"
   ```

4. Tunggu hingga layanan berhasil diterapkan.

   ```
   $ aws proton wait service-created --name "static-website"
   ```

5. Ambil output dan lihat situs web baru Anda.

   Jalankan perintah berikut:

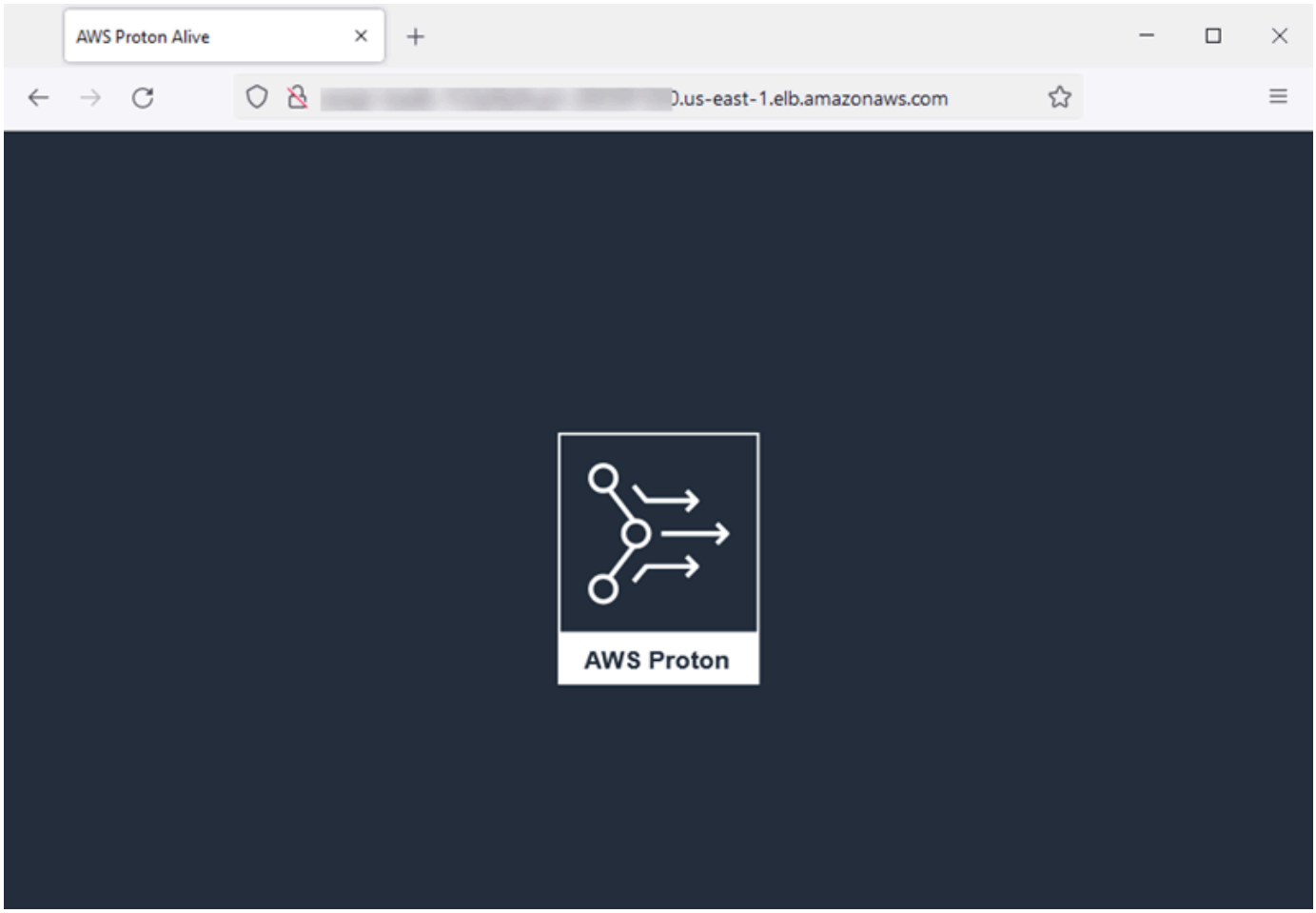
   ```
   $ aws proton list-service-instance-outputs \
     --service-name "static-website" \
     --service-instance-name load-balanced-fargate-svc-prod
   ```

   Output perintah harus mirip dengan yang berikut ini:

   ```
   {
       "outputs": [
           {
               "key": "ServiceURL",
   ```

```
            "valueString": "http://your-service-endpoint.us-
east-1.elb.amazonaws.com"
        }
    ]
}
```

Nilai output `ServiceURL` instance adalah titik akhir ke situs web layanan baru Anda. Gunakan browser Anda untuk menavigasi ke sana. Anda akan melihat grafik berikut pada halaman statis:



Langkah 5: Bersihkan (opsional)

Pada langkah ini, ketika Anda selesai menjelajahi AWS sumber daya yang Anda buat sebagai bagian dari tutorial ini, dan untuk menghemat biaya yang terkait dengan sumber daya ini, Anda menghapusnya.

Untuk menghapus sumber daya tutorial

1. Untuk menghapus layanan, jalankan perintah berikut:

```
$ aws proton delete-service --name "static-website"
```

2. Untuk menghapus lingkungan, jalankan perintah berikut:

```
$ aws proton delete-environment --name "fargate-env-prod"
```

3. Untuk menghapus template layanan, jalankan perintah berikut:

```
$ aws proton delete-template-sync-config \
  --template-name "load-balanced-fargate-svc" \
  --template-type "SERVICE"
$ aws proton delete-service-template --name "load-balanced-fargate-svc"
```

4. Untuk menghapus template lingkungan, jalankan perintah berikut:

```
$ aws proton delete-template-sync-config \
  --template-name "fargate-env" \
  --template-type "ENVIRONMENT"
$ aws proton delete-environment-template --name "fargate-env"
```

Pustaka AWS Proton template

AWS Proton Tim memelihara pustaka contoh template pada GitHub. Pustaka mencakup contoh file infrastruktur sebagai kode (IAc) untuk banyak skenario infrastruktur lingkungan dan aplikasi umum.

Pustaka template disimpan dalam dua GitHub repositori:

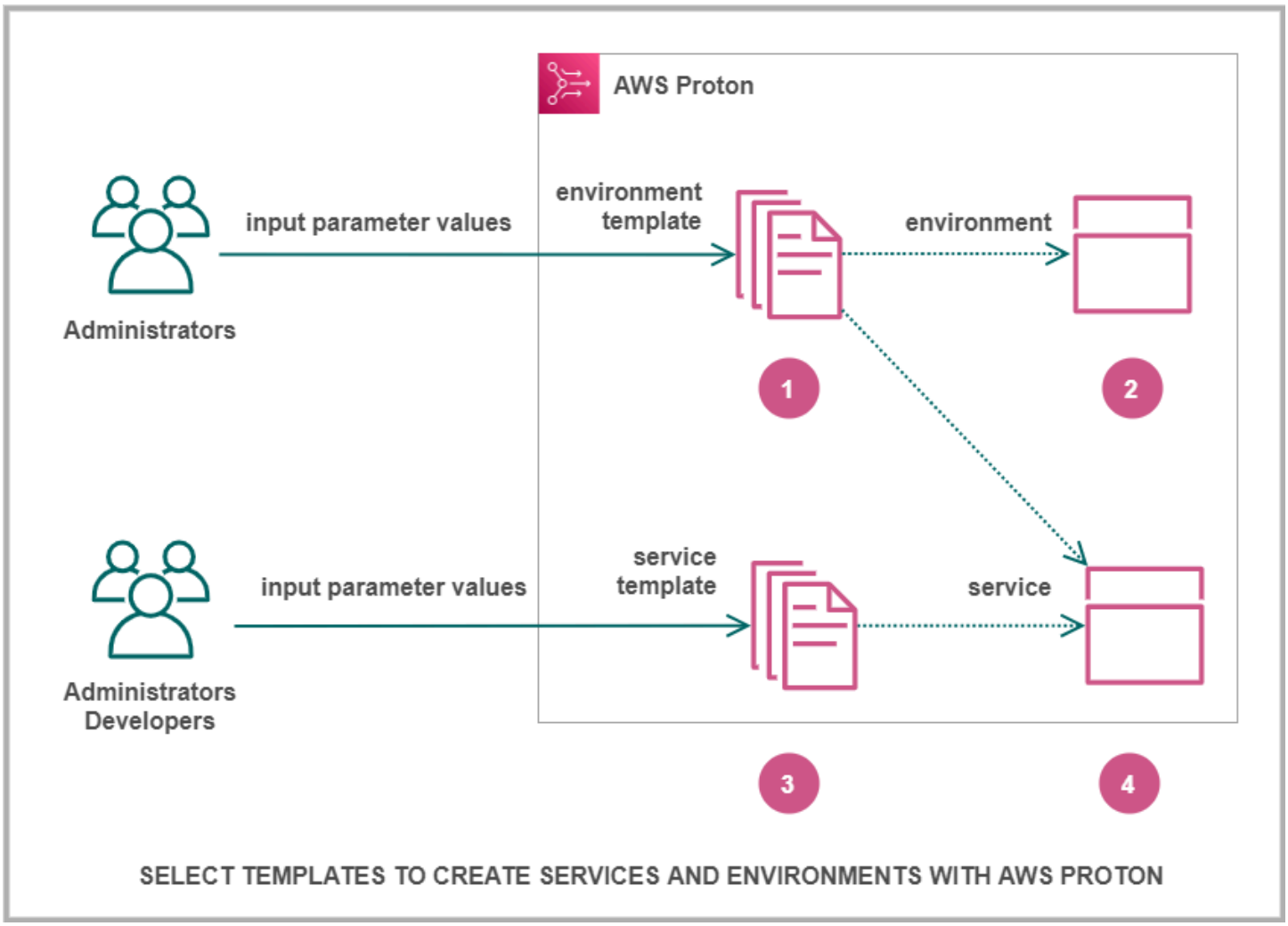
- aws-proton-cloudformation-sample-templates — Contoh bundel template yang digunakan AWS CloudFormationdengan Jinja sebagai bahasa IAc mereka. Anda dapat menggunakan contoh-contoh ini untuk AWS-penyediaan terkelola lingkungan.
- aws-proton-terraform-sample-templates — Contoh bundel template yang menggunakan Terraform sebagai bahasa IAc mereka. Anda dapat menggunakan contoh-contoh ini untuk Penyediaan yang dikelola sendiri lingkungan.

Masing-masing repositori ini memiliki `README` file dengan informasi lengkap tentang konten dan struktur repositori. Setiap contoh memiliki informasi tentang kasus penggunaan yang dicakup template, arsitektur contoh, dan parameter input yang diambil template.

Anda dapat menggunakan templat di perpustakaan ini secara langsung dengan mem-forking salah satu repositori perpustakaan ke akun Anda. GitHub Atau, gunakan contoh-contoh ini sebagai titik awal untuk mengembangkan lingkungan dan template layanan Anda.

Bagaimana cara AWS Proton kerja

Dengan AWS Proton, Anda menyediakan lingkungan, dan kemudian layanan yang berjalan di lingkungan tersebut. Lingkungan dan layanan didasarkan pada template lingkungan dan layanan, masing-masing, yang Anda pilih di pustaka templat AWS Proton berversi Anda.



1 Ketika Anda, sebagai administrator, memilih template lingkungan dengan AWS Proton, Anda memberikan nilai untuk parameter input yang diperlukan.

2 AWS Proton menggunakan template lingkungan dan nilai parameter untuk menyediakan lingkungan Anda.

3

Ketika Anda, sebagai pengembang atau administrator, memilih template layanan dengan AWS Proton, Anda memberikan nilai untuk parameter input yang diperlukan. Anda juga memilih lingkungan untuk menyebarkan aplikasi atau layanan Anda.

4

AWS Proton menggunakan templat layanan, dan nilai parameter layanan dan lingkungan yang dipilih, untuk menyediakan layanan Anda.

Anda memberikan nilai untuk parameter input untuk menyesuaikan template Anda untuk digunakan kembali dan beberapa kasus penggunaan, aplikasi, atau layanan.

Untuk membuat ini berfungsi, Anda membuat bundel template lingkungan atau layanan dan mengunggahnya ke lingkungan terdaftar atau templat layanan, masing-masing.

Bundel template berisi semua yang AWS Proton dibutuhkan untuk menyediakan lingkungan atau layanan.

Saat membuat template lingkungan atau layanan, Anda mengunggah bundel templat yang berisi file parametrized infrastructure as code (IAc) yang AWS Proton digunakan untuk menyediakan lingkungan atau layanan.

Saat Anda memilih template lingkungan atau layanan untuk membuat atau memperbarui lingkungan atau layanan, Anda memberikan nilai untuk parameter file iAc bundel template.

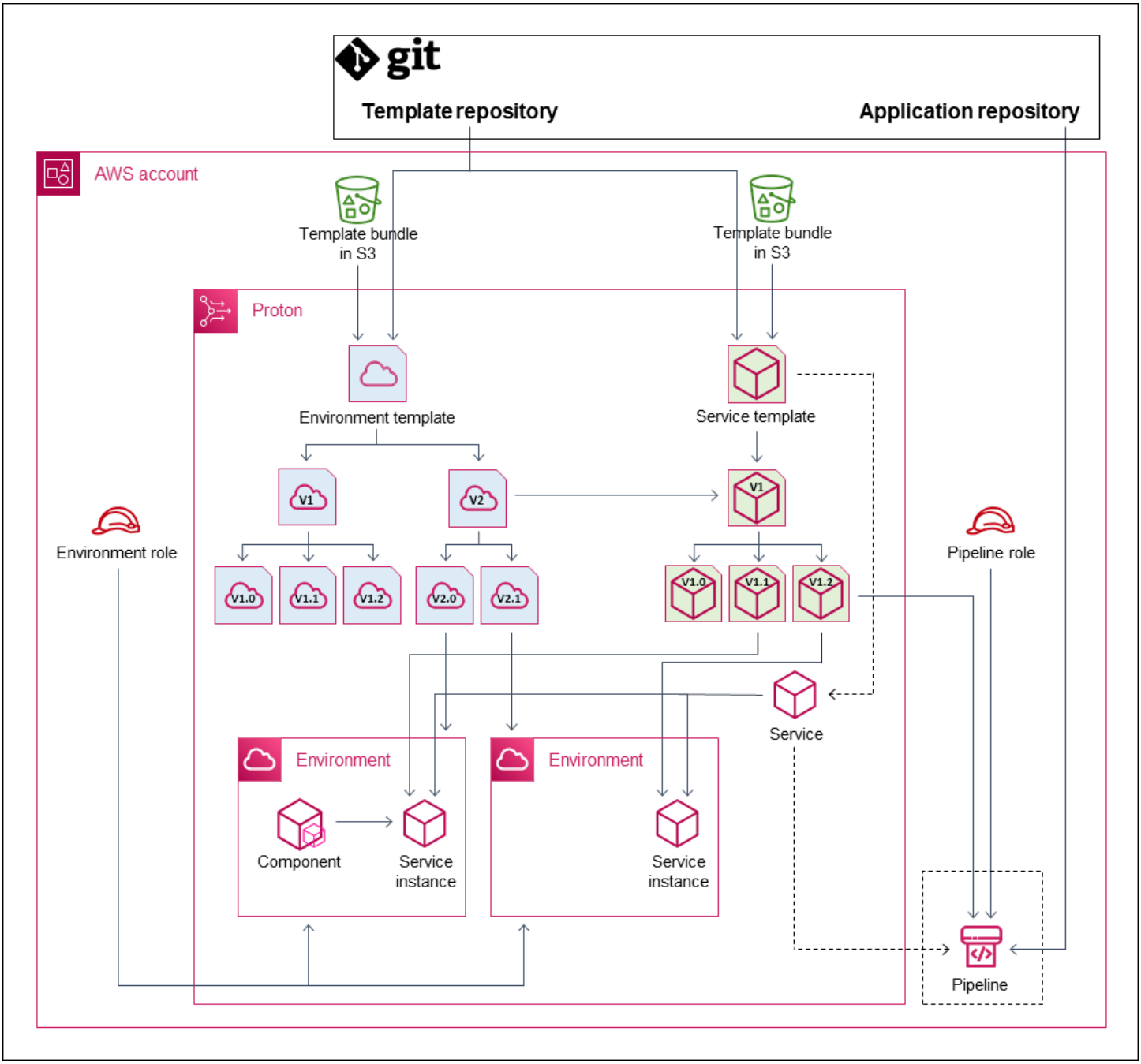
Topik

- AWS Proton benda-benda
- Bagaimana AWS Proton ketentuan infrastruktur
- AWS Proton terminologi

AWS Proton benda-benda

Diagram berikut menunjukkan AWS Proton objek utama dan hubungannya dengan objek lain AWS dan pihak ketiga. Panah mewakili arah aliran data (arah ketergantungan terbalik).

Kami mengikuti diagram dengan deskripsi singkat dan tautan referensi untuk AWS Proton objek-objek ini.



- Template lingkungan - Kumpulan versi template lingkungan yang dapat digunakan untuk membuat AWS Proton lingkungan.

  Untuk informasi selengkapnya, lihat Penulisan templat dan bundel dan Template.

- Versi template lingkungan - Versi spesifik dari template lingkungan. Mengambil bundel template sebagai input, baik dari bucket S3 atau dari repositori Git. Bundel menentukan Infrastructure as Code (IAc) dan parameter input terkait untuk suatu AWS Proton lingkungan.

Lihat informasi selengkapnya di the section called “Versi”, the section called “Publikasikan”, dan the section called “Konfigurasi sinkronisasi templat”.

- Lingkungan - Kumpulan sumber daya AWS infrastruktur bersama dan kebijakan akses yang digunakan AWS Proton layanan. AWS sumber daya disediakan dengan menggunakan versi template lingkungan yang dipanggil dengan nilai parameter tertentu. Kebijakan akses disediakan dalam peran layanan.

  Untuk informasi selengkapnya, lihat Lingkungan.

- Template layanan — Kumpulan versi template layanan yang dapat digunakan untuk membuat AWS Proton layanan.

  Untuk informasi selengkapnya, lihat Penulisan templat dan bundel dan Template.

- Versi template layanan - Versi spesifik dari template layanan. Mengambil bundel template sebagai input, baik dari bucket S3 atau dari repositori Git. Bundel menentukan Infrastructure as Code (IAc) dan parameter input terkait untuk suatu AWS Proton layanan.

  Versi template layanan juga menentukan batasan ini pada instance layanan berdasarkan versi:

  - Template lingkungan yang kompatibel - Instans hanya dapat berjalan di lingkungan berdasarkan templat lingkungan yang kompatibel ini.
  - Sumber komponen yang didukung — Jenis komponen yang dapat diasosiasikan pengembang dengan instance.

  Lihat informasi selengkapnya di the section called “Versi”, the section called “Publikasikan”, dan the section called “Konfigurasi sinkronisasi templat”.

- Service - Kumpulan instance layanan yang menjalankan aplikasi menggunakan sumber daya yang ditentukan dalam template layanan, dan opsional CI/CD pipeline yang menyebarkan kode aplikasi ke instance ini.

  Dalam diagram, garis putus-putus dari template Layanan berarti bahwa layanan meneruskan template ke instance layanan dan pipeline.

  Untuk informasi selengkapnya, lihat Layanan .

- Service instance — Kumpulan sumber daya AWS infrastruktur yang menjalankan aplikasi di AWS Proton lingkungan tertentu. AWS sumber daya disediakan dengan menggunakan versi templat layanan yang dipanggil dengan nilai parameter tertentu.

  Untuk informasi selengkapnya, lihat Layanan dan the section called “Perbarui contoh”.

- Pipeline — CI/CD Pipeline opsional yang menyebarkan aplikasi ke dalam instance layanan, dengan kebijakan akses untuk menyediakan pipeline ini. Kebijakan akses disediakan dalam peran layanan. Layanan tidak selalu memiliki AWS Proton pipeline terkait—Anda dapat memilih untuk mengelola penerapan kode aplikasi di luar. AWS Proton

  Dalam diagram, garis putus-putus dari Layanan dan kotak putus-putus di sekitar Pipeline berarti bahwa jika Anda memilih untuk mengelola CI/CD penerapan sendiri, AWS Proton pipeline mungkin tidak dibuat, dan pipeline Anda sendiri mungkin tidak berada dalam akun Anda. AWS

  Untuk informasi selengkapnya, lihat Layanan dan the section called “Perbarui pipa”.
- Komponen - Ekstensi yang ditentukan pengembang untuk instance layanan. Menentukan sumber daya AWS infrastruktur tambahan yang mungkin dibutuhkan aplikasi tertentu, selain sumber daya yang disediakan oleh lingkungan dan instance layanan. Tim platform mengontrol infrastruktur yang dapat disediakan komponen dengan melampirkan peran komponen ke lingkungan.

  Untuk informasi selengkapnya, lihat Komponen-komponen.

Bagaimana AWS Proton ketentuan infrastruktur

AWS Proton dapat menyediakan infrastruktur dengan salah satu dari beberapa cara:

- AWS-managed provisioning — AWS Proton memanggil mesin penyediaan atas nama Anda. Metode ini hanya mendukung bundel AWS CloudFormation template. Untuk informasi selengkapnya, lihat the section called “CloudFormation File iAc”.
- CodeBuild provisioning — AWS Proton digunakan AWS CodeBuild untuk menjalankan perintah shell yang Anda berikan. Perintah Anda dapat membaca input yang AWS Proton menyediakan, dan bertanggung jawab untuk penyediaan atau deprovisioning infrastruktur dan menghasilkan nilai output. Bundel template untuk metode ini menyertakan perintah Anda dalam file manifes dan program, skrip, atau file lain yang mungkin diperlukan oleh perintah ini.

  Sebagai contoh untuk menggunakan CodeBuild penyediaan, Anda dapat menyertakan kode yang menggunakan AWS sumber daya AWS Cloud Development Kit (AWS CDK) untuk menyediakan, dan manifes yang menginstal CDK dan menjalankan kode CDK Anda.

  Untuk informasi selengkapnya, lihat the section called “CodeBuild bundel”.

> **Note**
>
> Anda dapat menggunakan CodeBuild penyediaan dengan lingkungan dan layanan. Saat ini Anda tidak dapat menyediakan komponen dengan cara ini.

- Penyediaan yang dikelola sendiri — AWS Proton mengeluarkan permintaan tarik (PR) ke repositori yang Anda sediakan, tempat sistem penyebaran infrastruktur Anda sendiri menjalankan proses penyediaan. Metode ini hanya mendukung bundel template Terraform. Untuk informasi selengkapnya, lihat the section called “File Terraform IAc”.

AWS Proton menentukan dan menetapkan metode penyediaan untuk setiap lingkungan dan layanan secara terpisah. Saat Anda membuat atau memperbarui lingkungan atau layanan, AWS Proton periksa bundel templat yang Anda berikan, dan tentukan metode penyediaan yang ditunjukkan oleh bundel templat. Pada tingkat lingkungan, Anda memberikan parameter yang mungkin dibutuhkan lingkungan dan layanan potensial untuk metode penyediaannya—AWS Identity and Access Management (IAM) peran, koneksi akun lingkungan, atau repositori infrastruktur.

Pengembang yang menggunakan AWS Proton untuk menyediakan layanan memiliki pengalaman yang sama terlepas dari metode penyediaan. Pengembang tidak perlu mengetahui metode penyediaan dan tidak perlu mengubah apa pun dalam proses penyediaan layanan. Templat layanan menetapkan metode penyediaan, dan setiap lingkungan yang digunakan pengembang layanan untuk menyediakan parameter yang diperlukan untuk penyediaan instance layanan.

Diagram berikut merangkum beberapa ciri utama dari metode penyediaan yang berbeda. Bagian yang mengikuti tabel memberikan rincian tentang setiap metode.

Metode penyediaan	Template	Disediakan oleh	Status dilacak oleh
AWS-dikelola	manifes, skema, file IAc () CloudFormation	AWS Proton (melalui CloudFormation)	AWS Proton (melalui CloudFormation)
CodeBuild	manifes (dengan perintah), skema, dependensi perintah (misalnya kode) AWS CDK	AWS Proton (melalui CodeBuild)	AWS Proton (perintah Anda mengembal ikan status melalui CodeBuild)

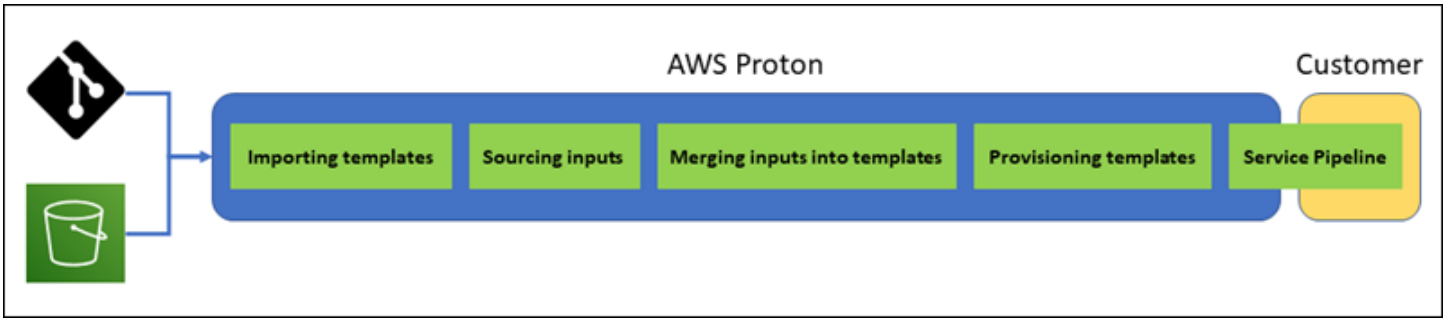
Metode penyediaan	Template	Disediakan oleh	Status dilacak oleh
dikelola sendiri	manifes, skema, file IAc (Terraform)	Kode Anda (melalui tindakan Git)	Kode Anda (diteruskan AWS melalui panggilan API)

Cara kerja AWS-managed provisioning

Ketika lingkungan atau layanan menggunakan AWS-managed provisioning, infrastruktur disediakan sebagai berikut:

1. AWS Proton Pelanggan (administrator atau pengembang) menciptakan AWS Proton sumber daya (lingkungan atau layanan). Pelanggan memilih templat untuk sumber daya dan menyediakan parameter yang diperlukan. Untuk informasi lebih lanjut, lihat bagian berikut,the section called “Pertimbangan untuk penyediaan AWS-managed”.
2. AWS Proton membuat CloudFormation template lengkap untuk menyediakan sumber daya.
3. AWS Proton panggilan CloudFormation untuk memulai penyediaan menggunakan template yang dirender.
4. AWS Proton terus memantau CloudFormation penyebaran.
5. Saat penyediaan selesai, AWS Proton laporkan kembali kesalahan jika terjadi kegagalan, dan menangkap output penyediaan, seperti ID VPC Amazon, jika berhasil.

Diagram berikut menunjukkan bahwa AWS Proton menangani sebagian besar langkah-langkah ini secara langsung.



Pertimbangan untuk penyediaan AWS-managed

- Peran penyediaan infrastruktur — Ketika lingkungan atau salah satu instance layanan yang berjalan di dalamnya mungkin menggunakan AWS-managed provisioning, administrator perlu mengonfigurasi peran IAM (baik secara langsung atau sebagai bagian dari koneksi akun lingkungan). AWS Proton AWS Proton menggunakan peran ini untuk menyediakan infrastruktur sumber daya penyediaan AWS yang dikelola ini. Peran harus memiliki izin untuk digunakan CloudFormation untuk membuat semua sumber daya yang disertakan dalam templat sumber daya ini.

  Untuk informasi selengkapnya, lihat the section called “Peran IAM” dan the section called “Contoh kebijakan peran layanan”.
- Penyediaan layanan — Saat pengembang menerapkan instance layanan yang menggunakan AWS-managed provisioning ke lingkungan, AWS Proton gunakan peran yang diberikan ke lingkungan tersebut untuk menyediakan infrastruktur untuk instance layanan. Pengembang tidak melihat peran ini dan tidak dapat mengubahnya.
- Service with pipeline — Template layanan yang menggunakan AWS-managed provisioning dapat menyertakan definisi pipeline yang ditulis dalam skema YAMB. CloudFormation AWS Proton juga membuat pipeline dengan menelepon CloudFormation. Peran yang AWS Proton digunakan untuk membuat pipa terpisah dari peran untuk setiap lingkungan individu. Peran ini disediakan secara AWS Proton terpisah, hanya sekali di tingkat AWS akun, dan digunakan untuk menyediakan dan mengelola semua saluran pipa yang AWS dikelola. Peran ini harus memiliki izin untuk membuat saluran pipa dan sumber daya lain yang dibutuhkan saluran pipa Anda.

  Prosedur berikut menunjukkan bagaimana memberikan peran pipa ke AWS Proton.

  AWS Proton console

  Untuk memberikan peran pipa

  1. Di AWS Proton konsol, pada panel navigasi, pilih Pengaturan > Pengaturan akun, lalu pilih Konfigurasi.
  2. Gunakan bagian Pipeline AWS-managed role untuk mengonfigurasi peran pipeline baru atau yang sudah ada untuk penyediaan AWS-managed.

AWS Proton API

Untuk memberikan peran pipa

1. Gunakan tindakan UpdateAccountSettingsAPI.
2. Berikan Nama Sumber Daya Amazon (ARN) peran layanan pipeline Anda dalam parameter. `pipelineServiceRoleArn`

AWS CLI

Untuk memberikan peran pipa

Jalankan perintah berikut:

```
$ aws proton update-account-settings \
    --pipeline-service-role-arn \
        "arn:aws:iam::123456789012:role/my-pipeline-role"
```

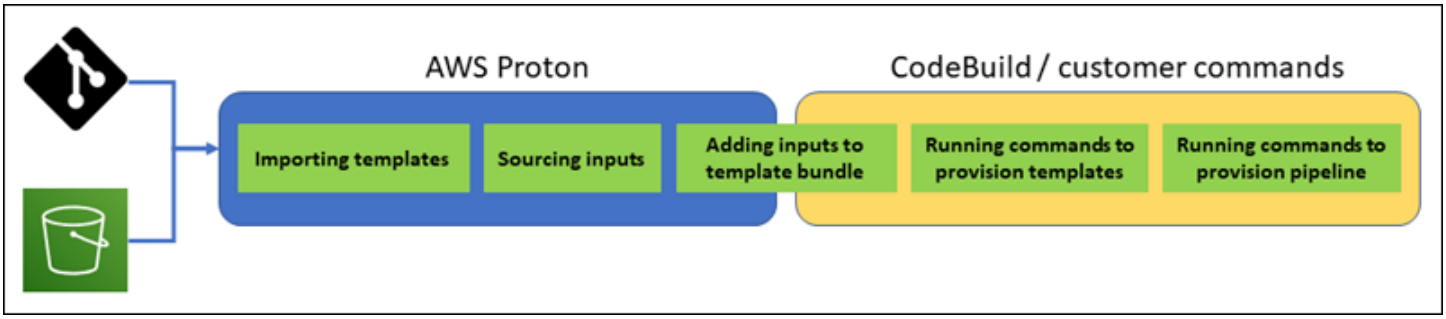
Cara CodeBuild kerja penyediaan

Ketika lingkungan atau layanan menggunakan CodeBuild penyediaan, infrastruktur disediakan sebagai berikut:

1. AWS Proton Pelanggan (administrator atau pengembang) menciptakan AWS Proton sumber daya (lingkungan atau layanan). Pelanggan memilih templat untuk sumber daya dan menyediakan parameter yang diperlukan. Untuk informasi lebih lanjut, lihat bagian berikut,the section called "Pertimbangan untuk penyediaan CodeBuild".
2. AWS Proton merender file input dengan nilai parameter masukan untuk penyediaan sumber daya.
3. AWS Proton panggilan CodeBuild untuk memulai pekerjaan. CodeBuild Pekerjaan menjalankan perintah shell pelanggan yang ditentukan dalam template. Perintah ini menyediakan infrastruktur yang diinginkan, sementara secara opsional membaca nilai input.
4. Saat penyediaan selesai, perintah pelanggan akhir mengembalikan status penyediaan CodeBuild dan memanggil tindakan NotifyResourceDeploymentStatusChange AWS Proton API untuk memberikan output, seperti ID VPC Amazon, jika ada.

> **Important**
>
> Pastikan bahwa perintah Anda mengembalikan status penyediaan dengan benar CodeBuild dan memberikan output. Jika tidak, tidak AWS Proton dapat melacak status penyediaan dengan benar dan tidak dapat memberikan output yang benar ke instance layanan.

Diagram berikut menggambarkan langkah-langkah yang AWS Proton dilakukan dan langkah-langkah yang dilakukan perintah Anda dalam suatu CodeBuild pekerjaan.



Pertimbangan untuk penyediaan CodeBuild

- Peran penyediaan infrastruktur - Ketika lingkungan atau salah satu instance layanan yang berjalan di dalamnya mungkin menggunakan penyediaan CodeBuild berbasis, administrator perlu mengonfigurasi peran IAM (baik secara langsung atau sebagai bagian dari koneksi akun lingkungan). AWS Proton AWS Proton menggunakan peran ini untuk menyediakan infrastruktur sumber daya CodeBuild penyediaan ini. Peran harus memiliki izin untuk digunakan CodeBuild untuk membuat semua sumber daya yang Anda perintahkan dalam templat penyediaan sumber daya ini.

  Untuk informasi selengkapnya, lihat the section called “Peran IAM” dan the section called “Contoh kebijakan peran layanan”.
- Penyediaan layanan — Saat pengembang menerapkan instance layanan yang menggunakan CodeBuild penyediaan ke lingkungan, AWS Proton gunakan peran yang disediakan ke lingkungan tersebut untuk menyediakan infrastruktur untuk instance layanan. Pengembang tidak melihat peran ini dan tidak dapat mengubahnya.
- Service with pipeline — Template layanan yang menggunakan CodeBuild provisioning dapat menyertakan perintah untuk menyediakan pipeline. AWS Proton juga membuat pipeline dengan

menelepon CodeBuild. Peran yang AWS Proton digunakan untuk membuat pipa terpisah dari peran untuk setiap lingkungan individu. Peran ini diberikan secara AWS Proton terpisah, hanya sekali di tingkat AWS akun, dan digunakan untuk menyediakan dan mengelola semua pipeline CodeBuild berbasis. Peran ini harus memiliki izin untuk membuat saluran pipa dan sumber daya lain yang dibutuhkan saluran pipa Anda.

Prosedur berikut menunjukkan bagaimana memberikan peran pipa ke AWS Proton.

AWS Proton console

Untuk memberikan peran pipa

1. Di AWS Proton konsol, pada panel navigasi, pilih Pengaturan > Pengaturan akun, lalu pilih Konfigurasi.
2. Gunakan bagian peran penyediaan pipeline Codebuild untuk mengonfigurasi peran pipeline baru atau yang sudah ada untuk penyediaan. CodeBuild

AWS Proton API

Untuk memberikan peran pipa

1. Gunakan tindakan UpdateAccountSettingsAPI.
2. Berikan Nama Sumber Daya Amazon (ARN) peran layanan pipeline Anda dalam parameter. `pipelineCodebuildRoleArn`

AWS CLI

Untuk memberikan peran pipa

Jalankan perintah berikut:

```
$ aws proton update-account-settings \
    --pipeline-codebuild-role-arn \
        "arn:aws:iam::123456789012:role/my-pipeline-role"
```

Cara kerja penyediaan yang dikelola sendiri

Ketika lingkungan dikonfigurasi untuk menggunakan penyediaan yang dikelola sendiri, infrastruktur disediakan sebagai berikut:

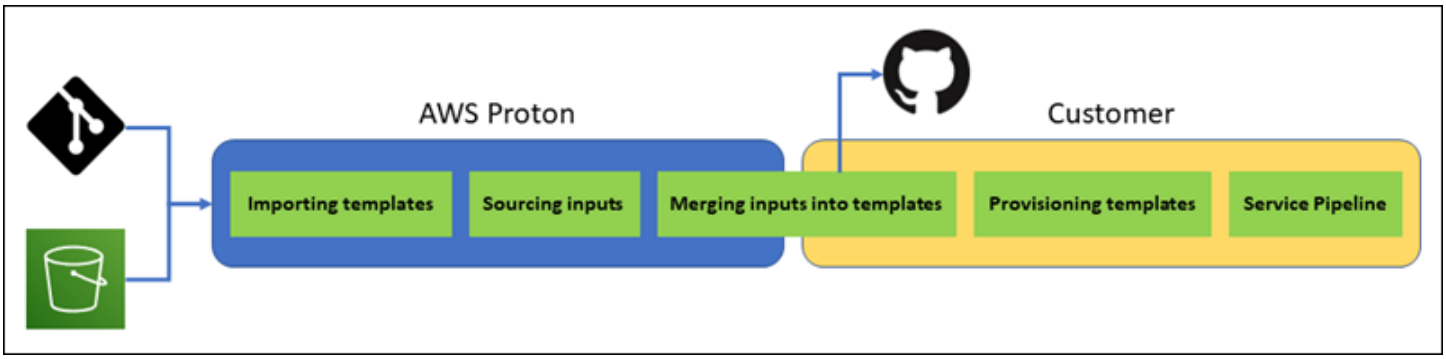
1. AWS Proton Pelanggan (administrator atau pengembang) menciptakan AWS Proton sumber daya (lingkungan atau layanan). Pelanggan memilih templat untuk sumber daya dan menyediakan parameter yang diperlukan. Untuk lingkungan, pelanggan juga menyediakan repositori infrastruktur terkait. Untuk informasi lebih lanjut, lihat bagian berikut,the section called “Pertimbangan untuk penyediaan yang dikelola sendiri”.
2. AWS Proton membuat template Terraform lengkap. Ini terdiri dari satu atau lebih file Terraform, berpotensi dalam beberapa folder, dan file `.tfvars` variabel. AWS Proton menulis nilai parameter yang disediakan pada panggilan pembuatan sumber daya ke dalam file variabel ini.
3. AWS Proton mengirimkan PR ke repositori infrastruktur dengan template Terraform yang dirender.
4. Saat pelanggan (administrator atau pengembang) menggabungkan PR, otomatisasi pelanggan memicu mesin penyediaan untuk memulai penyediaan infrastruktur menggunakan templat gabungan.

   > **Note**
   >
   > Jika pelanggan (administrator atau pengembang) menutup PR, AWS Proton mengenali PR sebagai tertutup dan menandai penerapan sebagai dibatalkan.

5. Saat penyediaan selesai, otomatisasi pelanggan memanggil tindakan NotifyResourceDeploymentStatusChange AWS Proton API untuk menunjukkan penyelesaian, memberikan status (keberhasilan atau kegagalan), dan memberikan output, seperti ID VPC Amazon, jika ada.

   > **Important**
   >
   > Pastikan kode otomatisasi Anda memanggil kembali AWS Proton dengan status penyediaan dan output. Jika tidak, AWS Proton mungkin menganggap penyediaan sebagai tertunda lebih lama dari yang seharusnya, dan terus tampilkan status Dalam proses.

Diagram berikut menggambarkan langkah-langkah yang AWS Proton dilakukan dan langkah-langkah yang dilakukan oleh sistem penyediaan Anda sendiri.



Pertimbangan untuk penyediaan yang dikelola sendiri

- Repositori infrastruktur — Ketika administrator mengonfigurasi lingkungan untuk penyediaan yang dikelola sendiri, mereka perlu menyediakan repositori infrastruktur tertaut. AWS Proton mengirimkan PRs ke repositori ini untuk menyediakan infrastruktur lingkungan dan semua instance layanan yang digunakan untuk itu. Tindakan otomatisasi milik pelanggan di repositori harus mengambil peran IAM dengan izin untuk membuat semua sumber daya yang disertakan dalam templat lingkungan dan layanan Anda, dan identitas yang mencerminkan akun tujuan. AWS Untuk contoh GitHub Tindakan yang mengasumsikan peran, lihat Mengasumsikan Peran dalam dokumentasi Tindakan Untuk Tindakan “Konfigurasi AWS Kredensial”. GitHub
- Izin - Kode penyediaan Anda harus mengautentikasi dengan akun seperlunya (misalnya, mengautentikasi ke AWS akun) dan memberikan otorisasi penyediaan sumber daya (misalnya, memberikan peran).
- Penyediaan layanan — Saat pengembang menerapkan instance layanan yang menggunakan penyediaan yang dikelola sendiri ke lingkungan, AWS Proton mengirimkan PR ke repositori yang terkait dengan lingkungan untuk menyediakan infrastruktur untuk instance layanan. Pengembang tidak melihat repositori dan tidak dapat mengubahnya.

> **Note**
>
> Pengembang yang membuat layanan menggunakan proses yang sama terlepas dari metode penyediaan, dan perbedaannya diabstraksikan dari mereka. Namun, dengan penyediaan yang dikelola sendiri, pengembang mungkin mengalami respons yang lebih lambat, karena mereka perlu menunggu sampai seseorang (yang mungkin bukan diri mereka sendiri) menggabungkan PR dalam repositori infrastruktur sebelum penyediaan dapat dimulai.

- Layanan dengan pipa — Templat layanan untuk lingkungan dengan penyediaan yang dikelola sendiri dapat mencakup definisi pipa (misalnya, AWS CodePipeline pipa), yang ditulis dalam Terraform HCL. AWS Proton Untuk mengaktifkan penyediaan pipeline ini, administrator menyediakan repositori pipeline tertaut ke. AWS Proton Saat menyediakan pipeline, tindakan otomatisasi milik pelanggan di repositori harus mengambil peran IAM dengan izin untuk menyediakan pipeline, dan identitas yang mencerminkan akun tujuan. AWS Repositori dan peran pipa terpisah dari yang digunakan untuk setiap lingkungan individu. Repositori tertaut disediakan secara AWS Proton terpisah, hanya sekali di tingkat AWS akun, dan digunakan untuk menyediakan dan mengelola semua pipeline. Peran tersebut harus memiliki izin untuk membuat saluran pipa dan sumber daya lain yang dibutuhkan saluran pipa Anda.

  Prosedur berikut menunjukkan bagaimana menyediakan repositori dan peran pipa. AWS Proton

  AWS Proton console

  Untuk memberikan peran pipa

  1. Di AWS Proton konsol, pada panel navigasi, pilih Pengaturan > Pengaturan akun, lalu pilih Konfigurasi.
  2. Gunakan bagian repositori pipa CI/CD untuk mengonfigurasi tautan repositori baru atau yang sudah ada.

  AWS Proton API

  Untuk memberikan peran pipa

  1. Gunakan tindakan UpdateAccountSettingsAPI.
  2. Berikan penyedia, nama, dan cabang repositori pipeline Anda dalam parameter. `pipelineProvisioningRepository`

  AWS CLI

  Untuk memberikan peran pipa

  Jalankan perintah berikut:

  ```
  $ aws proton update-account-settings \
      --pipeline-provisioning-repository \
          "provider=GITHUB,name=my-pipeline-repo-name,branch=my-branch"
  ```

- Penghapusan sumber daya yang disediakan sendiri — Modul Terraform dapat mencakup elemen konfigurasi yang diperlukan untuk operasi Terraform, selain definisi sumber daya. Oleh karena itu, tidak AWS Proton dapat menghapus semua file Terraform untuk lingkungan atau instance layanan. Sebagai gantinya, AWS Proton tandai file untuk dihapus dan perbarui bendera di metadata PR. Otomatisasi Anda dapat membaca flag itu dan menggunakannya untuk memicu perintah terraform destroy.

AWS Proton terminologi

Template lingkungan

Mendefinisikan infrastruktur bersama, seperti VPC atau cluster, yang digunakan oleh beberapa aplikasi atau sumber daya.

Bundel template lingkungan

Kumpulan file yang Anda unggah untuk membuat dan mendaftarkan template lingkungan AWS Proton. Bundel template lingkungan berisi berikut ini:

1. File skema yang mendefinisikan infrastruktur sebagai parameter input kode.
2. File infrastruktur sebagai kode (IAc) yang mendefinisikan infrastruktur bersama, seperti VPC atau cluster, yang digunakan oleh beberapa aplikasi atau sumber daya.
3. File manifes yang mencantumkan file IAc.

Lingkungan

Infrastruktur bersama yang disediakan, seperti VPC atau cluster, yang digunakan oleh beberapa aplikasi atau sumber daya.

Template layanan

Mendefinisikan jenis infrastruktur yang diperlukan untuk menyebarkan dan memelihara aplikasi atau layanan mikro di lingkungan.

Paket template layanan

Kumpulan file yang Anda unggah untuk membuat dan mendaftarkan template layanan AWS Proton. Sebuah paket template layanan berisi berikut ini:

1. File skema yang mendefinisikan infrastruktur sebagai parameter input kode (IAc).
2. File IAC yang mendefinisikan infrastruktur yang diperlukan untuk menyebarkan dan memelihara aplikasi atau layanan mikro di lingkungan.

3. File manifes yang mencantumkan file IAc.
4. Opsional
   a. File IAC yang mendefinisikan infrastruktur pipa layanan.
   b. File manifes yang mencantumkan file IAc.

Layanan

Infrastruktur yang disediakan yang diperlukan untuk menyebarkan dan memelihara aplikasi atau layanan mikro di lingkungan.

Contoh layanan

Infrastruktur yang disediakan yang mendukung aplikasi atau layanan mikro di suatu lingkungan.

Pipa layanan

Infrastruktur yang disediakan yang mendukung pipa.

Versi template

Sebuah versi mayor atau minor dari template. Untuk informasi selengkapnya, lihat Template berversi.

Parameter input

Didefinisikan dalam file skema dan digunakan dalam file infrastruktur sebagai kode (IAc) sehingga file IAc dapat digunakan secara berulang dan untuk berbagai kasus penggunaan.

Berkas skema

Mendefinisikan infrastruktur sebagai parameter input file kode.

Berkas spesifikasi

Menentukan nilai untuk infrastruktur sebagai parameter input file kode, seperti yang didefinisikan dalam file skema.

Berkas manifes

Daftar infrastruktur sebagai file kode.

Membuat template dan membuat bundel untuk AWS Proton

AWS Proton menyediakan sumber daya untuk Anda berdasarkan infrastruktur sebagai file kode (IAc). Anda menjelaskan infrastruktur dalam file IAc yang dapat digunakan kembali. Untuk membuat file dapat digunakan kembali untuk lingkungan dan aplikasi yang berbeda, Anda menulisnya sebagai templat, menentukan parameter input, dan menggunakan parameter ini dalam definisi IAc. Saat nanti Anda membuat sumber daya penyediaan (lingkungan, instance layanan, atau komponen), AWS Proton gunakan mesin rendering, yang menggabungkan nilai input dengan templat untuk membuat file IAc yang siap disediakan.

Administrator membuat sebagian besar templat sebagai bundel templat, lalu mengunggah dan mendaftarkannya. AWS Proton Sisa halaman ini membahas bundel AWS Proton template ini. Komponen yang didefinisikan secara langsung adalah pengecualian — pengembang membuatnya dan menyediakan file template IAC secara langsung. Untuk informasi selengkapnya tentang komponen, lihatKomponen-komponen.

Topik

- Bundel template
- AWS Proton parameter
- AWS Proton infrastruktur sebagai file kode
- Berkas skema
- Bungkus file template untuk AWS Proton
- Pertimbangan bundel template

Bundel template

Sebagai administrator, Anda membuat dan mendaftarkan template dengan AWS Proton. Anda menggunakan template ini untuk membuat lingkungan dan layanan. Saat Anda membuat layanan, menyediakan AWS Proton , dan menyebarkan instance layanan ke lingkungan yang dipilih. Untuk informasi selengkapnya, lihat AWS Proton untuk tim platform.

Untuk membuat dan mendaftarkan templat AWS Proton, Anda mengunggah bundel templat yang berisi file infrastruktur sebagai kode (IAc) yang AWS Proton perlu disediakan dan lingkungan atau layanan.

Sebuah bundel template berisi berikut ini:

- File Infrastructure as code (IAc) dengan file YAML manifes yang mencantumkan file IAc.
- File YAMAL skema untuk definisi parameter input file IAC Anda.

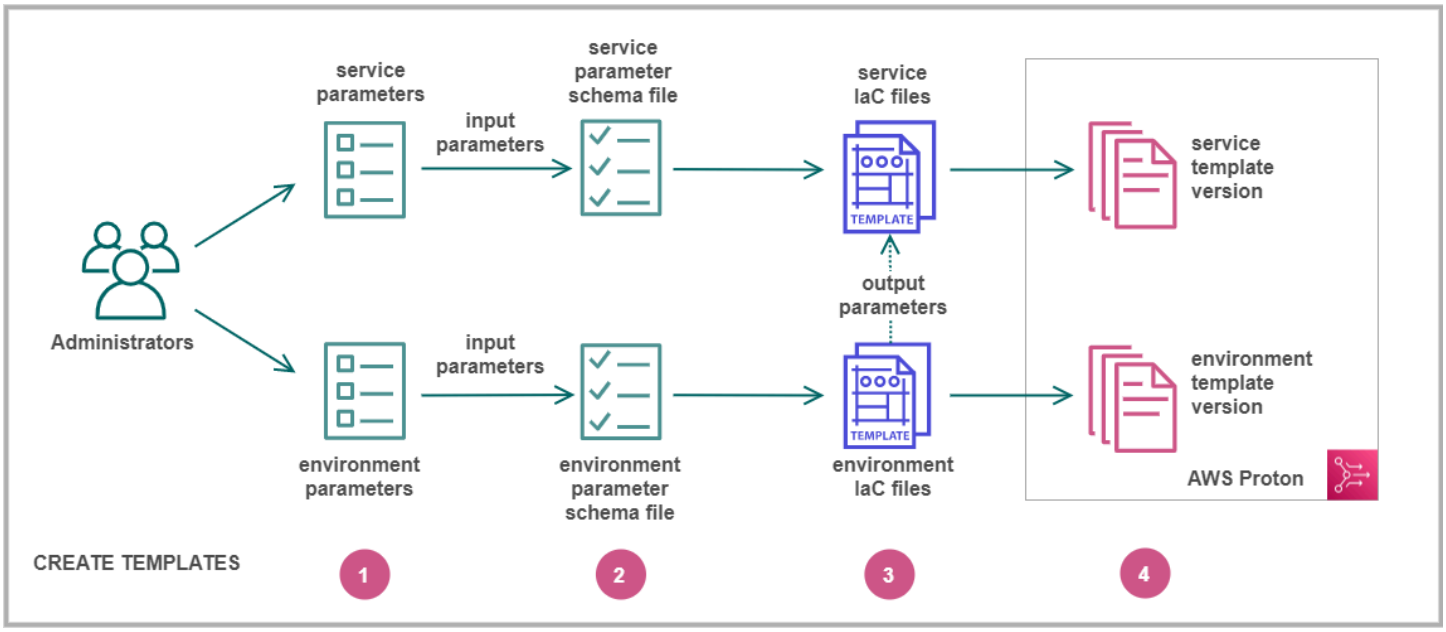
Bundel template CloudFormation lingkungan berisi satu file IAC.

Bundel template CloudFormation layanan berisi satu file IAc untuk definisi instance layanan dan file IAc opsional lainnya untuk definisi pipeline.

Lingkungan Terraform dan bundel template layanan masing-masing dapat berisi beberapa file IAc.

AWS Proton membutuhkan file skema parameter masukan. Saat Anda menggunakan AWS CloudFormation untuk membuat file IAC Anda, Anda menggunakan sintaks Jinja untuk mereferensikan parameter input Anda. AWS Proton menyediakan ruang nama parameter yang dapat Anda gunakan untuk mereferensikan parameter dalam file IAC Anda.

Diagram berikut menunjukkan contoh langkah-langkah yang dapat Anda ambil untuk membuat template AWS Proton.



1 Identifi parameter input.

2 Buat file skema untuk menentukan parameter input Anda.

3 Buat file IAC yang mereferensikan parameter input Anda. Anda dapat mereferensikan output file IAC lingkungan sebagai input untuk file iAC layanan Anda.

4 Daftark versi template dengan AWS Proton dan unggah bundel template Anda.

AWS Proton parameter

Anda dapat menentukan dan menggunakan parameter dalam infrastruktur Anda sebagai file kode (IAc) untuk membuatnya fleksibel dan dapat digunakan kembali. Anda membaca nilai parameter dalam file IAC Anda dengan mengacu pada nama parameter di namespace AWS Proton parameter. AWS Proton menyuntikkan nilai parameter ke dalam file IAc yang dirender yang dihasilkannya selama penyediaan sumber daya. Untuk memproses parameter AWS CloudFormation IAc, AWS Proton gunakan Jinja. Untuk memproses parameter Terraform IAc, AWS Proton buat file nilai parameter Terraform dan bergantung pada kemampuan parametriisasi yang dibangun ke dalam HCL.

DenganCodeBuild penyediaan, AWS Proton menghasilkan file input yang dapat diimpor kode Anda. File tersebut adalah file JSON atau HCL, tergantung pada properti di manifes template Anda. Untuk informasi selengkapnya, lihat the section called “CodeBuild parameter penyediaan”.

Anda dapat merujuk ke parameter di lingkungan, layanan, dan file iAc komponen atau kode penyediaan dengan persyaratan berikut:

- Panjang setiap nama parameter tidak melebihi 100 karakter.
- Panjang namespace parameter dan nama sumber daya digabungkan tidak melebihi batas karakter untuk nama sumber daya.

AWS Proton penyediaan gagal jika kuota ini terlampaui.

Jenis parameter

Jenis parameter berikut tersedia untuk Anda untuk referensi dalam file AWS Proton IAC:

Parameter masukan

Lingkungan dan instance layanan dapat mengambil parameter input yang Anda tentukan dalam file skema yang Anda kaitkan dengan template lingkungan atau layanan. Anda dapat merujuk ke parameter input sumber daya dalam file IAC sumber daya. File komponen IAC dapat merujuk ke parameter input dari instance layanan tempat komponen dilampirkan.

AWS Proton memeriksa nama parameter input terhadap file skema Anda, dan mencocokkannya dengan parameter yang direferensikan dalam file IAC Anda untuk menyuntikkan nilai input yang Anda berikan dalam file spesifikasi selama penyediaan sumber daya.

Parameter keluaran

Anda dapat menentukan output di salah satu file IAC Anda. Output dapat berupa, misalnya, nama, ID, atau ARN dari salah satu sumber daya yang disediakan template, atau dapat menjadi cara untuk melewati salah satu input template. Anda dapat merujuk ke output ini dalam file IAC dari sumber daya lain.

Dalam file CloudFormation IAC, tentukan parameter output di `Outputs:` blok. Dalam file Terraform IAc, tentukan setiap parameter output menggunakan pernyataan. `output`

Parameter sumber daya

AWS Proton secara otomatis membuat parameter AWS Proton sumber daya. Parameter ini mengekspos properti dari objek AWS Proton sumber daya. Contoh parameter sumber daya adalah`environment.name`.

Menggunakan AWS Proton parameter dalam file IAC Anda

Untuk membaca nilai parameter dalam file IAc, Anda merujuk ke nama parameter di namespace AWS Proton parameter. Untuk file AWS CloudFormation IAc, Anda menggunakan sintaks Jinja dan mengelilingi parameter dengan pasangan kurawal kurawal dan tanda kutip.

Tabel berikut menunjukkan sintaks referensi untuk setiap bahasa template yang didukung, dengan contoh.

Bahasa template	Sintaksis	Contoh: masukan lingkungan bernama “VPC”
CloudFormation	`"{{ parameter-name }}"`	`"{{ environment.inputs.VPC }}"`
Terraform	`var.parameter-name`	`var.environment.inputs.VPC` Definisi variabel Terraform yang dihasilkan

> **Note**
>
> Jika Anda menggunakan parameter CloudFormation dinamis dalam file IAC Anda, Anda harus menghindarinya untuk mencegah kesalahan salah tafsir Jinja. Untuk informasi selengkapnya, lihat Pemecahan masalah AWS Proton

Tabel berikut mencantumkan nama namespace untuk semua parameter AWS Proton sumber daya. Setiap jenis file template dapat menggunakan subset yang berbeda dari namespace parameter.

Berkas templat	Jenis paramete	Nama parameter	Deskripsi
Lingkungan	sumber daya	`environment.` **`name`**	Nama lingkungan
	input	`environment.inputs.` *`input-name`*	Masukan lingkungan yang ditentukan skema
Layanan	sumber daya	`environment.` **`name`** `environment.` **`account_id`**	Nama dan Akun AWS ID lingkungan
	output	`environment.outputs.` *`output-name`*	Output file iAc lingkungan
	sumber daya	`service.`**`branch_name`**	Nama layanan dan repositori kode

Berkas templat	Jenis paramete	Nama parameter	Deskripsi
		`service.`**`name`** `service.`**`repository_connection_arn`** `service.`**`repository_id`**	
	sumber daya	`service_instance.` **`name`**	Nama contoh layanan
	input	`service_instance.inputs.` *`input-name`*	Masukan contoh layanan yang ditentukan skema
	sumber daya	`service_instance.components.default.` **`name`**	Nama komponen default terlampir
	output	`service_instance.components.default.outputs.` *`output-name`*	Output file iAc komponen default terlampir
Alur	sumber daya	`service_instance.environment.` **`name`** `service_instance.environment.` **`account_id`**	Nama dan Akun AWS ID lingkungan instance layanan
	output	`service_instance.environment.outputs.` *`output-name`*	Output file iAc lingkungan contoh layanan
	input	`pipeline.inputs.` *`input-name`*	Input pipa yang ditentukan skema

Berkas templat	Jenis paramete	Nama parameter	Deskripsi
	sumber daya	`service.`**`branch_name`** `service.`**`name`** `service.`**`repository_connection_arn`** `service.`**`repository_id`**	Nama layanan dan repositori kode
	input	`service_instance.inputs.` *`input-name`*	Masukan contoh layanan yang ditentukan skema
	koleksi	`{% for service_instance in` **`service_instances`** `%}...{% endfor %}`	Kumpulan instance layanan yang dapat Anda lalui
Komponen	sumber daya	`environment.` **`name`** `environment.` **`account_id`**	Nama lingkungan dan ID Akun AWS akun
	output	`environment.outputs.` *`output-name`*	Output file iAc lingkungan
	sumber daya	`service.`**`branch_name`** `service.`**`name`** `service.`**`repository_connection_arn`** `service.`**`repository_id`**	Nama layanan dan repositori kode (komponen terlampir)
	sumber daya	`service_instance.` **`name`**	Nama instance layanan (komponen terlampir)

Berkas templat	Jenis paramete	Nama parameter	Deskripsi
	input	`service_instance.inputs.` *`input-name`*	Input instance layanan yang ditentukan skema (komponen terlampir)
	sumber daya	`component.` **`name`**	Nama komponen

Untuk informasi dan contoh selengkapnya, lihat subtopik tentang parameter dalam file templat IAC untuk berbagai jenis sumber daya dan bahasa templat.

Topik

- Rincian dan contoh parameter file CloudFormation IAC lingkungan
- Detail parameter file layanan CloudFormation IAC dan contoh
- Detail dan contoh parameter file CloudFormation iAc komponen
- Filter parameter untuk CloudFormation file IAc
- CodeBuild rincian parameter penyediaan dan contoh
- Infrastruktur Terraform sebagai detail dan contoh parameter file kode (IAc)

Rincian dan contoh parameter file CloudFormation IAC lingkungan

Anda dapat menentukan dan mereferensikan parameter di infrastruktur lingkungan Anda sebagai file kode (IAc). Untuk penjelasan rinci tentang AWS Proton parameter, jenis parameter, namespace parameter, dan cara menggunakan parameter dalam file IAC Anda, lihat. the section called “Parameter”

Tentukan parameter lingkungan

Anda dapat menentukan parameter input dan output untuk file iAC lingkungan.

- Parameter input - Tentukan parameter input lingkungan dalam file skema Anda.

  Daftar berikut mencakup contoh parameter input lingkungan untuk kasus penggunaan umum.

  - Nilai VPC CIDR

- Pengaturan penyeimbang beban
- Pengaturan basis data
- Batas waktu pemeriksaan kesehatan

Sebagai administrator, Anda dapat memberikan nilai untuk parameter masukan saat membuat lingkungan:

- Gunakan konsol untuk mengisi formulir berbasis skema yang AWS Proton menyediakan.
- Gunakan CLI untuk memberikan spesifikasi yang menyertakan nilai.

- Parameter keluaran — Tentukan output lingkungan di file iAC lingkungan Anda. Anda kemudian dapat merujuk ke output ini dalam file IAC dari sumber daya lain.

Baca nilai parameter dalam file iAc lingkungan

Anda dapat membaca parameter yang terkait dengan lingkungan di lingkungan file IAC. Anda membaca nilai parameter dengan mereferensikan nama parameter di namespace AWS Proton parameter.

- Parameter masukan — Baca nilai input lingkungan dengan referensi`environment.inputs.`*`input-name`*.
- Parameter sumber daya — Baca parameter AWS Proton sumber daya dengan mereferensikan nama seperti`environment.name`.

> **Note**
>
> Tidak ada parameter output dari sumber daya lain yang tersedia untuk file iAC lingkungan.

Contoh lingkungan dan layanan file IAc dengan parameter

Contoh berikut menunjukkan definisi parameter dan referensi dalam file iAc lingkungan. Contoh kemudian menunjukkan bagaimana parameter keluaran lingkungan yang didefinisikan dalam file iAc lingkungan dapat direferensikan dalam file iAc layanan.

Example File CloudFormation iAc lingkungan

Perhatikan hal berikut dalam contoh ini:

- `environment.inputs.`Namespace mengacu pada parameter input lingkungan.
- Parameter Amazon EC2 Systems Manager (SSM) `StoreInputValue` menggabungkan input lingkungan.
- `MyEnvParameterValue`Output memperlihatkan rangkaian parameter input yang sama sebagai parameter output. Tiga parameter output tambahan juga mengekspos parameter input secara individual.
- Enam parameter output tambahan mengekspos sumber daya yang disediakan lingkungan.

```
Resources:
  StoreInputValue:
    Type: AWS::SSM::Parameter
    Properties:
      Type: String
      Value: "{{ environment.inputs.my_sample_input }}
 {{ environment.inputs.my_other_sample_input}}
 {{ environment.inputs.another_optional_input }}"
              # input parameter references

# These output values are available to service infrastructure as code files as outputs,
 when given the
# the 'environment.outputs' namespace, for example,
 service_instance.environment.outputs.ClusterName.
Outputs:
  MyEnvParameterValue:                                          # output definition
    Value: !GetAtt StoreInputValue.Value
  MySampleInputValue:                                           # output definition
    Value: "{{ environment.inputs.my_sample_input }}"         #   input parameter
 reference
  MyOtherSampleInputValue:                                      # output definition
    Value: "{{ environment.inputs.my_other_sample_input }}"   #   input parameter
 reference
  AnotherOptionalInputValue:                                    # output definition
    Value: "{{ environment.inputs.another_optional_input }}"  #   input parameter
 reference
  ClusterName:                                                  # output definition
    Description: The name of the ECS cluster
    Value: !Ref 'ECSCluster'                                    #   provisioned resource
  ECSTaskExecutionRole:                                         # output definition
    Description: The ARN of the ECS role
    Value: !GetAtt 'ECSTaskExecutionRole.Arn'                   #   provisioned resource
  VpcId:                                                        # output definition
```

```
    Description: The ID of the VPC that this stack is deployed in
    Value: !Ref 'VPC'                                          #   provisioned resource
  PublicSubnetOne:                                             # output definition
    Description: Public subnet one
    Value: !Ref 'PublicSubnetOne'                              #   provisioned resource
  PublicSubnetTwo:                                             # output definition
    Description: Public subnet two
    Value: !Ref 'PublicSubnetTwo'                              #   provisioned resource
  ContainerSecurityGroup:                                      # output definition
    Description: A security group used to allow Fargate containers to receive traffic
    Value: !Ref 'ContainerSecurityGroup'                       #   provisioned resource
```

Example Layanan file CloudFormation iAc

`environment.outputs.`Namespace mengacu pada output lingkungan dari file iAc lingkungan. Misalnya, nama `environment.outputs.ClusterName` membaca nilai parameter keluaran `ClusterName` lingkungan.

```
AWSTemplateFormatVersion: '2010-09-09'
Description: Deploy a service on AWS Fargate, hosted in a public subnet, and accessible
 via a public load balancer.
Mappings:
  TaskSize:
    x-small:
      cpu: 256
      memory: 512
    small:
      cpu: 512
      memory: 1024
    medium:
      cpu: 1024
      memory: 2048
    large:
      cpu: 2048
      memory: 4096
    x-large:
      cpu: 4096
      memory: 8192
Resources:
  # A log group for storing the stdout logs from this service's containers
  LogGroup:
    Type: AWS::Logs::LogGroup
    Properties:
```

```
      LogGroupName: '{{service_instance.name}}' # resource parameter

  # The task definition. This is a simple metadata description of what
  # container to run, and what resource requirements it has.
  TaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      Family: '{{service_instance.name}}' # resource parameter
      Cpu: !FindInMap [TaskSize, {{service_instance.inputs.task_size}}, cpu] # input
parameter
      Memory: !FindInMap [TaskSize, {{service_instance.inputs.task_size}}, memory]
      NetworkMode: awsvpc
      RequiresCompatibilities:
        - FARGATE
      ExecutionRoleArn: '{{environment.outputs.ECSTaskExecutionRole}}'  # output
reference to an environment infrastructure code file
      TaskRoleArn: !Ref "AWS::NoValue"
      ContainerDefinitions:
        - Name: '{{service_instance.name}}'  # resource parameter
          Cpu: !FindInMap [TaskSize, {{service_instance.inputs.task_size}}, cpu]
          Memory: !FindInMap [TaskSize, {{service_instance.inputs.task_size}}, memory]
          Image: '{{service_instance.inputs.image}}'
          PortMappings:
            - ContainerPort: '{{service_instance.inputs.port}}' # input parameter
          LogConfiguration:
            LogDriver: 'awslogs'
            Options:
              awslogs-group: '{{service_instance.name}}' # resource parameter
              awslogs-region: !Ref 'AWS::Region'
              awslogs-stream-prefix: '{{service_instance.name}}' # resource parameter

  # The service_instance. The service is a resource which allows you to run multiple
  # copies of a type of task, and gather up their logs and metrics, as well
  # as monitor the number of running tasks and replace any that have crashed
  Service:
    Type: AWS::ECS::Service
    DependsOn: LoadBalancerRule
    Properties:
      ServiceName: '{{service_instance.name}}'  # resource parameter
      Cluster: '{{environment.outputs.ClusterName}}' # output reference to an
environment infrastructure as code file
      LaunchType: FARGATE
      DeploymentConfiguration:
        MaximumPercent: 200
```

```
        MinimumHealthyPercent: 75
      DesiredCount: '{{service_instance.inputs.desired_count}}'# input parameter
      NetworkConfiguration:
        AwsvpcConfiguration:
          AssignPublicIp: ENABLED
          SecurityGroups:
            - '{{environment.outputs.ContainerSecurityGroup}}' # output reference to an
 environment infrastructure as code file
          Subnets:
            - '{{environment.outputs.PublicSubnetOne}}' # output reference to an
 environment infrastructure as code file
            - '{{environment.outputs.PublicSubnetTwo}}' # output reference to an
 environment infrastructure as code file
      TaskDefinition: !Ref 'TaskDefinition'
      LoadBalancers:
        - ContainerName: '{{service_instance.name}}'  # resource parameter
          ContainerPort: '{{service_instance.inputs.port}}' # input parameter
          TargetGroupArn: !Ref 'TargetGroup'
[...]
```

Detail parameter file layanan CloudFormation IAC dan contoh

Anda dapat menentukan dan mereferensikan parameter dalam layanan dan infrastruktur pipa sebagai file kode (IAc). Untuk penjelasan rinci tentang AWS Proton parameter, jenis parameter, namespace parameter, dan cara menggunakan parameter dalam file IAC Anda, lihat. the section called “Parameter”

Tentukan parameter layanan

Anda dapat menentukan parameter input dan output untuk file iAC layanan.

- Parameter input - Tentukan parameter input instance layanan dalam file skema Anda.

  Daftar berikut mencakup contoh parameter input layanan untuk kasus penggunaan umum.

  - Port
  - Ukuran tugas
  - Citra
  - Jumlah yang diinginkan
  - Berkas Docker
  - Perintah uji unit

Anda memberikan nilai untuk parameter masukan saat membuat layanan:

- Gunakan konsol untuk mengisi formulir berbasis skema yang AWS Proton menyediakan.
- Gunakan CLI untuk memberikan spesifikasi yang menyertakan nilai.

- Parameter keluaran — Tentukan output instance layanan dalam file iAC layanan Anda. Anda kemudian dapat merujuk ke output ini dalam file IAC dari sumber daya lain.

Baca nilai parameter dalam file iAc layanan

Anda dapat membaca parameter yang terkait dengan layanan dan sumber daya lain dalam file iAC layanan. Anda membaca nilai parameter dengan mereferensikan nama parameter di namespace AWS Proton parameter.

- Parameter input — Baca nilai input instance layanan dengan referensi`service_instance.inputs.`*`input-name`*.
- Parameter sumber daya — Baca parameter AWS Proton sumber daya dengan mereferensikan nama seperti`service.name`,`service_instance.name`, dan`environment.name`.
- Parameter keluaran — Baca output sumber daya lain dengan referensi `environment.outputs.`*`output-name`* atau. `service_instance.components.default.outputs.`*`output-name`*

Contoh layanan file iAc dengan parameter

Contoh berikut adalah cuplikan dari file layanan CloudFormation IAc. `environment.outputs.`Namespace mengacu pada output dari file iAc lingkungan. `service_instance.inputs.`Namespace mengacu pada parameter input instance layanan. `service_instance.name`Properti mengacu pada parameter AWS Proton sumber daya.

```
Resources:
  StoreServiceInstanceInputValue:
    Type: AWS::SSM::Parameter
    Properties:
      Type: String
      Value: "{{ service.name }} {{ service_instance.name }}
 {{ service_instance.inputs.my_sample_service_instance_required_input }}
 {{ service_instance.inputs.my_sample_service_instance_optional_input }}
 {{ environment.outputs.MySampleInputValue }}
 {{ environment.outputs.MyOtherSampleInputValue }}"
```

```
              #  resource parameter references               # input parameter
 references
                                            # output references to an environment
 infrastructure as code file
Outputs:
  MyServiceInstanceParameter:                                                    #
 output definition
    Value: !Ref StoreServiceInstanceInputValue
  MyServiceInstanceRequiredInputValue:                                           #
 output definition
    Value: "{{ service_instance.inputs.my_sample_service_instance_required_input }}"  #
 input parameter reference
  MyServiceInstanceOptionalInputValue:                                           #
 output definition
    Value: "{{ service_instance.inputs.my_sample_service_instance_optional_input }}"  #
 input parameter reference
  MyServiceInstancesEnvironmentSampleOutputValue:                                #
 output definition
    Value: "{{ environment.outputs.MySampleInputValue }}"                        #
 output reference to an environment IaC file
  MyServiceInstancesEnvironmentOtherSampleOutputValue:                           #
 output definition
    Value: "{{ environment.outputs.MyOtherSampleInputValue }}"                   #
 output reference to an environment IaC file
```

Detail dan contoh parameter file CloudFormation iAc komponen

Anda dapat menentukan dan mereferensikan parameter dalam infrastruktur komponen Anda sebagai file kode (IAc). Untuk penjelasan rinci tentang AWS Proton parameter, jenis parameter, namespace parameter, dan cara menggunakan parameter dalam file IAC Anda, lihat. the section called “Parameter” Untuk informasi selengkapnya tentang komponen, lihatKomponen-komponen.

Tentukan parameter keluaran komponen

Anda dapat menentukan parameter output dalam file iAc komponen Anda. Anda kemudian dapat merujuk ke output ini dalam file layanan IAC.

> **Note**
>
> Anda tidak dapat menentukan input untuk file komponen IAC. Komponen terlampir bisa mendapatkan masukan dari instance layanan yang mereka lampirkan. Komponen terpisah tidak memiliki input.

Baca nilai parameter dalam file iAc komponen

Anda dapat membaca parameter yang terkait dengan komponen dan sumber daya lain dalam file komponen IAc. Anda membaca nilai parameter dengan mereferensikan nama parameter di namespace AWS Proton parameter.

- Parameter input — Baca nilai input instance layanan terlampir dengan referensi`service_instance.inputs.`*`input-name`*.
- Parameter sumber daya — Baca parameter AWS Proton sumber daya dengan mereferensikan nama seperti`component.name`,`service.name`,`service_instance.name`, dan`environment.name`.
- Parameter keluaran — Baca output lingkungan dengan referensi`environment.outputs.`*`output-name`*.

Contoh komponen dan layanan file IAc dengan parameter

Contoh berikut menunjukkan komponen yang menyediakan bucket Amazon Simple Storage Service (Amazon S3) dan kebijakan akses terkait serta mengekspos Amazon Resource Names ARNs () dari kedua sumber daya sebagai output komponen. Template Service IAC menambahkan output komponen sebagai variabel lingkungan container dari tugas Amazon Elastic Container Service (Amazon ECS) untuk membuat output tersedia untuk kode yang berjalan di container, dan menambahkan kebijakan akses bucket ke peran tugas. Nama bucket didasarkan pada nama lingkungan, layanan, instance layanan, dan komponen, yang berarti bahwa bucket digabungkan dengan instance spesifik dari template komponen yang memperluas instance layanan tertentu. Pengembang dapat membuat beberapa komponen kustom berdasarkan templat komponen ini, untuk menyediakan bucket Amazon S3 untuk berbagai instans layanan dan kebutuhan fungsional.

Contoh menunjukkan bagaimana Anda menggunakan `{{ ... }}` sintaks Jinja untuk merujuk ke komponen dan parameter sumber daya lainnya dalam file iAc layanan Anda. Anda dapat menggunakan `{% if ... %}` pernyataan untuk menambahkan blok pernyataan hanya ketika

komponen dilampirkan ke instance layanan. `proton_cfn_*`Kata kunci adalah filter yang dapat Anda gunakan untuk membersihkan dan memformat nilai parameter output. Untuk informasi lebih lanjut tentang filter, lihat the section called “CloudFormation filter parameter”.

Sebagai administrator, Anda membuat file template layanan IAC.

Example layanan file CloudFormation iAc menggunakan komponen

```
# service/instance_infrastructure/cloudformation.yaml

Resources:
  TaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      TaskRoleArn: !Ref TaskRole
      ContainerDefinitions:
        - Name: '{{service_instance.name}}'
          # ...
          {% if service_instance.components.default.outputs | length > 0 %}
          Environment:
            {{ service_instance.components.default.outputs |
                proton_cfn_ecs_task_definition_formatted_env_vars }}
          {% endif %}

  # ...

  TaskRole:
    Type: AWS::IAM::Role
    Properties:
      # ...
      ManagedPolicyArns:
        - !Ref BaseTaskRoleManagedPolicy
        {{ service_instance.components.default.outputs
            | proton_cfn_iam_policy_arns }}

  # Basic permissions for the task
  BaseTaskRoleManagedPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      # ...
```

Sebagai pengembang, Anda membuat file template komponen IAC.

Example file komponen CloudFormation iAc

```
# cloudformation.yaml

# A component that defines an S3 bucket and a policy for accessing the bucket.
Resources:
  S3Bucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: '{{environment.name}}-{{service.name}}-{{service_instance.name}}-
{{component.name}}'
  S3BucketAccessPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      PolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Effect: Allow
            Action:
              - 's3:Get*'
              - 's3:List*'
              - 's3:PutObject'
            Resource: !GetAtt S3Bucket.Arn
Outputs:
  BucketName:
    Description: "Bucket to access"
    Value: !GetAtt S3Bucket.Arn
  BucketAccessPolicyArn:
    Value: !Ref S3BucketAccessPolicy
```

Saat AWS Proton merender CloudFormation template untuk instance layanan Anda dan mengganti semua parameter dengan nilai aktual, template mungkin terlihat seperti file berikut.

Example contoh layanan yang CloudFormation diberikan file IAc

```
Resources:
  TaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      TaskRoleArn: !Ref TaskRole
      ContainerDefinitions:
        - Name: '{{service_instance.name}}'
          # ...
```

```
          Environment:
            - Name: BucketName
              Value: arn:aws:s3:us-
east-1:123456789012:environment_name-service_name-service_instance_name-component_name
            - Name: BucketAccessPolicyArn
              Value: arn:aws:iam::123456789012:policy/cfn-generated-policy-name
  # ...

  TaskRole:
    Type: AWS::IAM::Role
    Properties:
      # ...
      ManagedPolicyArns:
        - !Ref BaseTaskRoleManagedPolicy
        - arn:aws:iam::123456789012:policy/cfn-generated-policy-name

  # Basic permissions for the task
  BaseTaskRoleManagedPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      # ...
```

Filter parameter untuk CloudFormation file IAc

Saat Anda membuat referensi ke AWS Proton parameter dalam file AWS CloudFormation IAC Anda, Anda dapat menggunakan pengubah Jinja yang dikenal sebagai filter untuk memvalidasi, memfilter, dan memformat nilai parameter sebelum dimasukkan ke dalam templat yang dirender. Validasi filter sangat berguna ketika mengacu pada parameter keluaran komponen, karena pembuatan dan lampiran komponen dilakukan oleh pengembang, dan administrator yang menggunakan output komponen dalam templat instance layanan mungkin ingin memverifikasi keberadaan dan validitasnya. Namun, Anda dapat menggunakan filter di file Jinja IAc apa pun.

Bagian berikut menjelaskan dan menentukan filter parameter yang tersedia, dan memberikan contoh. AWS Proton mendefinisikan sebagian besar filter ini. `default`Filternya adalah filter bawaan Jinja.

Memformat properti lingkungan untuk tugas Amazon ECS

Deklarasi

```
dict # proton_cfn_ecs_task_definition_formatted_env_vars (raw: boolean = True) # YAML
 list of dicts
```

Deskripsi

Filter ini memformat daftar output yang akan digunakan dalam properti Lingkungan di `ContainerDefinition` bagian definisi tugas Amazon Elastic Container Service (Amazon ECS) Service Elastic Container ECS).

Setel `raw` `False` untuk juga memvalidasi nilai parameter. Dalam hal ini, nilai diperlukan untuk mencocokkan ekspresi reguler`^[a-zA-Z0-9_-]*$`. Jika nilai gagal validasi ini, rendering template gagal.

Contoh

Dengan template komponen kustom berikut:

```
Resources:
  # ...
Outputs:
  Output1:
    Description: "Example component output 1"
    Value: hello
  Output2:
    Description: "Example component output 2"
    Value: world
```

Dan template layanan berikut:

```
Resources:
  TaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      # ...
      ContainerDefinitions:
        - Name: MyServiceName
          # ...
          Environment:
            {{ service_instance.components.default.outputs
              | proton_cfn_ecs_task_definition_formatted_env_vars }}
```

Template layanan yang diberikan adalah sebagai berikut:

```
Resources:
```

```
  TaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      # ...
      ContainerDefinitions:
        - Name: MyServiceName
          # ...
          Environment:
            - Name: Output1
              Value: hello
            - Name: Output2
              Value: world
```

Format properti lingkungan untuk fungsi Lambda

Deklarasi

```
dict # proton_cfn_lambda_function_formatted_env_vars (raw: boolean = True) # YAML dict
```

Deskripsi

Filter ini memformat daftar output yang akan digunakan dalam properti Lingkungan di `Properties` bagian definisi AWS Lambda fungsi.

Setel `raw` `False` untuk juga memvalidasi nilai parameter. Dalam hal ini, nilai diperlukan untuk mencocokkan ekspresi reguler`^[a-zA-Z0-9_-]*$`. Jika nilai gagal validasi ini, rendering template gagal.

Contoh

Dengan template komponen kustom berikut:

```
Resources:
  # ...
Outputs:
  Output1:
    Description: "Example component output 1"
    Value: hello
  Output2:
    Description: "Example component output 2"
    Value: world
```

Dan template layanan berikut:

```
Resources:
  Lambda:
    Type: AWS::Lambda::Function
    Properties:
      Environment:
        Variables:
          {{ service_instance.components.default.outputs
            | proton_cfn_lambda_function_formatted_env_vars }}
```

Template layanan yang diberikan adalah sebagai berikut:

```
Resources:
  Lambda:
    Type: AWS::Lambda::Function
    Properties:
      Environment:
        Variables:
          Output1: hello
          Output2: world
```

Ekstrak kebijakan IAM ARNs untuk dimasukkan dalam peran IAM

Deklarasi

```
dict # proton_cfn_iam_policy_arns # YAML list
```

Deskripsi

Filter ini memformat daftar output yang akan digunakan dalam ManagedPolicyArns properti di `Properties` bagian definisi peran AWS Identity and Access Management (IAM). Filter menggunakan ekspresi reguler `^arn:[a-zA-Z-]+:iam::\d{12}:policy/` untuk mengekstrak kebijakan IAM yang valid ARNs dari daftar parameter keluaran. Anda dapat menggunakan filter ini untuk menambahkan kebijakan dalam nilai parameter keluaran ke definisi peran IAM dalam templat layanan.

Contoh

Dengan template komponen kustom berikut:

```
Resources:
  # ...
  ExamplePolicy1:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      # ...
  ExamplePolicy2:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      # ...

  # ...

Outputs:
  Output1:
    Description: "Example component output 1"
    Value: hello
  Output2:
    Description: "Example component output 2"
    Value: world
  PolicyArn1:
    Description: "ARN of policy 1"
    Value: !Ref ExamplePolicy1
  PolicyArn2:
    Description: "ARN of policy 2"
    Value: !Ref ExamplePolicy2
```

Dan template layanan berikut:

```
Resources:

  # ...

  TaskRole:
    Type: AWS::IAM::Role
    Properties:
      # ...
      ManagedPolicyArns:
        - !Ref BaseTaskRoleManagedPolicy
        {{ service_instance.components.default.outputs
            | proton_cfn_iam_policy_arns }}

  # Basic permissions for the task
```

```
  BaseTaskRoleManagedPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      # ...
```

Template layanan yang diberikan adalah sebagai berikut:

```
Resources:

  # ...

  TaskRole:
    Type: AWS::IAM::Role
    Properties:
      # ...
      ManagedPolicyArns:
        - !Ref BaseTaskRoleManagedPolicy
        - arn:aws:iam::123456789012:policy/cfn-generated-policy-name-1
        - arn:aws:iam::123456789012:policy/cfn-generated-policy-name-2

  # Basic permissions for the task
  BaseTaskRoleManagedPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      # ...
```

Sanitasi nilai properti

Deklarasi

```
string # proton_cfn_sanitize # string
```

Deskripsi

Ini adalah filter tujuan umum. Gunakan untuk memvalidasi keamanan nilai parameter. Filter memvalidasi bahwa nilainya cocok dengan ekspresi reguler `^[a-zA-Z0-9_-]*$` atau merupakan Nama Sumber Daya Amazon (ARN) yang valid. Jika nilai gagal validasi ini, rendering template gagal.

Contoh

Dengan template komponen kustom berikut:

```
Resources:
  # ...
Outputs:
  Output1:
    Description: "Example of valid output"
    Value: "This-is_valid_37"
  Output2:
    Description: "Example incorrect output"
    Value: "this::is::incorrect"
  SomeArn:
    Description: "Example ARN"
    Value: arn:aws:some-service::123456789012:some-resource/resource-name
```

- Referensi berikut dalam template layanan:

```
# ...
  {{ service_instance.components.default.outputs.Output1
    | proton_cfn_sanitize }}
```

  Render sebagai berikut:

```
# ...
  This-is_valid_37
```

- Referensi berikut dalam template layanan:

```
# ...
  {{ service_instance.components.default.outputs.Output2
    | proton_cfn_sanitize }}
```

  Hasil dengan kesalahan rendering berikut:

```
Illegal character(s) detected in "this::is::incorrect". Must match regex ^[a-zA-
Z0-9_-]*$ or be a valid ARN
```

- Referensi berikut dalam template layanan:

```
# ...
  {{ service_instance.components.default.outputs.SomeArn
    | proton_cfn_sanitize }}
```

Render sebagai berikut:

```
# ...
  arn:aws:some-service::123456789012:some-resource/resource-name
```

Berikan nilai default untuk referensi yang tidak ada

Deskripsi

`default`Filter memberikan nilai default ketika referensi namespace tidak ada. Gunakan untuk menulis template yang kuat yang dapat dirender tanpa kegagalan bahkan ketika parameter yang Anda rujuk hilang.

Contoh

Referensi berikut dalam template layanan menyebabkan rendering template gagal jika instance layanan tidak memiliki komponen terlampir yang didefinisikan secara langsung (default), atau jika komponen terlampir tidak memiliki output bernama`test`.

```
# ...
  {{ service_instance.components.default.outputs.test }}
```

Untuk menghindari masalah ini, tambahkan `default` filter.

```
# ...
  {{ service_instance.components.default.outputs.test | default("[optional-value]") }}
```

CodeBuild rincian parameter penyediaan dan contoh

Anda dapat menentukan parameter dalam template untuk AWS Proton sumber daya CodeBuild berbasis dan mereferensikan parameter ini dalam kode penyediaan Anda. Untuk penjelasan rinci tentang AWS Proton parameter, jenis parameter, namespace parameter, dan cara menggunakan parameter dalam file IAC Anda, lihat. the section called “Parameter”

> **Note**
>
> Anda dapat menggunakan CodeBuild penyediaan dengan lingkungan dan layanan. Saat ini Anda tidak dapat menyediakan komponen dengan cara ini.

Parameter input

Saat Anda membuat AWS Proton sumber daya, seperti lingkungan atau layanan, Anda memberikan nilai untuk parameter masukan yang ditentukan dalam file skema template Anda. Saat sumber daya yang Anda buat menggunakanCodeBuild penyediaan, AWS Proton merender nilai input ini ke dalam file input. Kode penyediaan Anda dapat mengimpor dan mendapatkan nilai parameter dari file ini.

Untuk contoh CodeBuild templat, lihatthe section called “CodeBuild bundel”. Untuk informasi selengkapnya tentang file manifes, lihatthe section called “Manifestasikan dan bungkus”.

Contoh berikut adalah file input JSON yang dihasilkan selama penyediaan CodeBuild berbasis instance layanan.

Contoh: menggunakan CodeBuild penyediaan AWS CDK with

```
{
  "service_instance": {
    "name": "my-service-staging",
    "inputs": {
      "port": "8080",
      "task_size": "medium"
    }
  },
  "service": {
    "name": "my-service"
  },
  "environment": {
    "account_id": "123456789012",
    "name": "my-env-staging",
    "outputs": {
      "vpc-id": "hdh2323423"
    }
  }
}
```

Parameter output

Untuk mengkomunikasikan output penyediaan sumber daya kembali ke AWS Proton, kode penyediaan Anda dapat menghasilkan file JSON bernama `proton-outputs.json` dengan nilai untuk parameter keluaran yang ditentukan dalam file skema template Anda. Misalnya, `cdk deploy` perintah memiliki `--outputs-file` argumen yang menginstruksikan AWS CDK untuk

menghasilkan file JSON dengan output penyediaan. Jika sumber daya Anda menggunakan AWS CDK, tentukan perintah berikut dalam manifes CodeBuild template Anda:

```
aws proton notify-resource-deployment-status-change
```

AWS Proton mencari file JSON ini. Jika file ada setelah kode penyediaan Anda berhasil diselesaikan, AWS Proton baca nilai parameter keluaran darinya.

Infrastruktur Terraform sebagai detail dan contoh parameter file kode (IAc)

Anda dapat menyertakan variabel input Terraform dalam `variable.tf` file di bundel template Anda. Anda juga dapat membuat skema untuk membuat variabel AWS Proton terkelola. AWS Proton membuat variabel `.tf files` dari file skema Anda. Untuk informasi selengkapnya, lihat the section called “File Terraform IAc”.

Untuk mereferensikan AWS Proton variabel yang ditentukan skema di infrastruktur Anda`.tf files`, Anda menggunakan AWS Proton ruang nama yang ditampilkan di Parameter dan ruang nama untuk tabel Terraform IAc. Misalnya, Anda dapat menggunakan`var.environment.inputs.vpc_cidr`. Di dalam tanda kutip, kelilingi variabel-variabel ini dengan tanda kurung tunggal dan tambahkan tanda dolar di depan penjepit pertama (misalnya,). “`${var.environment.inputs.vpc_cidr}`”

Contoh berikut menunjukkan cara menggunakan ruang nama untuk menyertakan AWS Proton parameter dalam lingkungan. `.tf file`

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 3.0"
    }
  }
  // This tells terraform to store the state file in s3 at the location
  // s3://terraform-state-bucket/tf-os-sample/terraform.tfstate
  backend "s3" {
    bucket = "terraform-state-bucket"
    key    = "tf-os-sample/terraform.tfstate"
    region = "us-east-1"
  }
}
```

```
// Configure the AWS Provider
provider "aws" {
  region = "us-east-1"
  default_tags {
    tags = var.proton_tags
  }
}

resource "aws_ssm_parameter" "my_ssm_parameter" {
  name  = "my_ssm_parameter"
  type  = "String"
  // Use the Proton environment.inputs. namespace
  value = var.environment.inputs.ssm_parameter_value
}
```

AWS Proton infrastruktur sebagai file kode

Bagian utama dari bundel template adalah file infrastruktur sebagai kode (IAc) yang menentukan sumber daya infrastruktur dan properti yang ingin Anda sediakan. AWS CloudFormation dan infrastruktur lainnya sebagai mesin kode menggunakan jenis file ini untuk menyediakan sumber daya infrastruktur.

> **Note**
>
> File IAC juga dapat digunakan secara independen dari bundel template, sebagai input langsung ke komponen yang didefinisikan secara langsung. Untuk informasi selengkapnya tentang komponen, lihatKomponen-komponen.

AWS Proton saat ini mendukung dua jenis file IAC:

- CloudFormationfile - Digunakan untuk penyediaan AWS-managed. AWS Proton menggunakan Jinja di atas format file CloudFormation template untuk parametrization.
- File HCL Terraform - Digunakan untuk penyediaan yang dikelola sendiri. HCL secara native mendukung parametriisasi.

Anda tidak dapat menyediakan AWS Proton sumber daya menggunakan kombinasi metode penyediaan. Anda harus menggunakan satu atau yang lain. Anda tidak dapat menerapkan layanan penyediaan yang AWS dikelola ke lingkungan penyediaan yang dikelola sendiri, atau sebaliknya.

Untuk informasi lebih lanjut, lihat the section called “Metode penyediaan”, Lingkungan, Layanan , dan Komponen-komponen .

CloudFormation File iAc

Pelajari cara menggunakan AWS CloudFormation infrastruktur sebagai file kode AWS Proton. CloudFormation adalah layanan infrastruktur sebagai kode (IAc) yang membantu Anda memodelkan dan mengatur AWS sumber daya Anda. Anda menentukan sumber daya infrastruktur Anda dalam template, menggunakan Jinja di atas format file CloudFormation template untuk parametrization. AWS Proton memperluas parameter dan membuat template lengkap CloudFormation . CloudFormation menyediakan sumber daya yang didefinisikan sebagai CloudFormation tumpukan. Untuk informasi selengkapnya, lihat Apa itu CloudFormation dalam CloudFormation Panduan Pengguna .

AWS Proton mendukung AWS-managed provisioning untuk IAc. CloudFormation

Mulailah dengan infrastruktur Anda sendiri yang ada sebagai file kode

Anda dapat mengadaptasi infrastruktur Anda sendiri yang ada sebagai file kode (IAc) untuk digunakan. AWS Proton

CloudFormation Contoh berikut, Contoh 1, dan Contoh 2, mewakili file CloudFormation IAc Anda sendiri yang ada. CloudFormation dapat menggunakan file-file ini untuk membuat dua CloudFormation tumpukan yang berbeda.

Dalam Contoh 1, file CloudFormation IAC dikonfigurasi untuk menyediakan infrastruktur untuk dibagikan di antara aplikasi kontainer. Dalam contoh ini, parameter input ditambahkan sehingga Anda dapat menggunakan file IAc yang sama untuk membuat beberapa set infrastruktur yang disediakan. Setiap set dapat memiliki nama yang berbeda bersama dengan set nilai VPC dan subnet CIDR yang berbeda. Baik sebagai administrator atau pengembang, Anda memberikan nilai untuk parameter ini saat Anda menggunakan file IAC untuk menyediakan sumber daya infrastruktur. CloudFormation Untuk kenyamanan Anda, parameter input ini ditandai dengan komentar dan direferensikan beberapa kali dalam contoh. Output didefinisikan di akhir template. Mereka dapat direferensikan dalam file CloudFormation IAC lainnya.

Dalam Contoh 2, file CloudFormation IAC dikonfigurasi untuk menyebarkan aplikasi ke infrastruktur yang disediakan dari Contoh 1. Parameter dikomentari untuk kenyamanan Anda.

Contoh 1: file CloudFormation IAc

```
AWSTemplateFormatVersion: '2010-09-09'
Description: AWS Fargate cluster running containers in a public subnet. Only supports
             public facing load balancer, and public service discovery namespaces.
Parameters:
   VpcCIDR:       # input parameter
        Description: CIDR for VPC
        Type: String
        Default: "10.0.0.0/16"
   SubnetOneCIDR: # input parameter
        Description: CIDR for SubnetOne
        Type: String
        Default: "10.0.0.0/24"
   SubnetTwoCIDR: # input parameters
        Description: CIDR for SubnetTwo
        Type: String
        Default: "10.0.1.0/24"
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      EnableDnsSupport: true
      EnableDnsHostnames: true
      CidrBlock:
        Ref: 'VpcCIDR'

  # Two public subnets, where containers will have public IP addresses
  PublicSubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      AvailabilityZone:
         Fn::Select:
         - 0
         - Fn::GetAZs: {Ref: 'AWS::Region'}
      VpcId: !Ref 'VPC'
      CidrBlock:
         Ref: 'SubnetOneCIDR'
      MapPublicIpOnLaunch: true

  PublicSubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      AvailabilityZone:
```

```
        Fn::Select:
        - 1
        - Fn::GetAZs: {Ref: 'AWS::Region'}
      VpcId: !Ref 'VPC'
      CidrBlock:
        Ref: 'SubnetTwoCIDR'
      MapPublicIpOnLaunch: true

  # Setup networking resources for the public subnets. Containers
  # in the public subnets have public IP addresses and the routing table
  # sends network traffic via the internet gateway.
  InternetGateway:
    Type: AWS::EC2::InternetGateway
  GatewayAttachement:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      VpcId: !Ref 'VPC'
      InternetGatewayId: !Ref 'InternetGateway'
  PublicRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref 'VPC'
  PublicRoute:
    Type: AWS::EC2::Route
    DependsOn: GatewayAttachement
    Properties:
      RouteTableId: !Ref 'PublicRouteTable'
      DestinationCidrBlock: '0.0.0.0/0'
      GatewayId: !Ref 'InternetGateway'
  PublicSubnetOneRouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref PublicSubnetOne
      RouteTableId: !Ref PublicRouteTable
  PublicSubnetTwoRouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref PublicSubnetTwo
      RouteTableId: !Ref PublicRouteTable

  # ECS Resources
  ECSCluster:
    Type: AWS::ECS::Cluster
```

```
  # A security group for the containers we will run in Fargate.
  # Rules are added to this security group based on what ingress you
  # add for the cluster.
  ContainerSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Access to the Fargate containers
      VpcId: !Ref 'VPC'

  # This is a role which is used by the ECS tasks themselves.
  ECSTaskExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
        - Effect: Allow
          Principal:
            Service: [ecs-tasks.amazonaws.com]
          Action: ['sts:AssumeRole']
      Path: /
      ManagedPolicyArns:
        - 'arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy'

# These output values will be available to other templates to use.
Outputs:
  ClusterName:                                          # output
    Description: The name of the ECS cluster
    Value: !Ref 'ECSCluster'
    Export:
      Name:
        Fn::Sub: "${AWS::StackName}-ECSCluster"
  ECSTaskExecutionRole:                                  # output
    Description: The ARN of the ECS role
    Value: !GetAtt 'ECSTaskExecutionRole.Arn'
    Export:
      Name:
        Fn::Sub: "${AWS::StackName}-ECSTaskExecutionRole"
  VpcId:                                                 # output
    Description: The ID of the VPC that this stack is deployed in
    Value: !Ref 'VPC'
    Export:
      Name:
        Fn::Sub: "${AWS::StackName}-VPC"
  PublicSubnetOne:                                       # output
```

```
    Description: Public subnet one
    Value: !Ref 'PublicSubnetOne'
    Export:
      Name:
        Fn::Sub: "${AWS::StackName}-PublicSubnetOne"
  PublicSubnetTwo:                                          # output
    Description: Public subnet two
    Value: !Ref 'PublicSubnetTwo'
    Export:
      Name:
        Fn::Sub: "${AWS::StackName}-PublicSubnetTwo"
  ContainerSecurityGroup:                                   # output
    Description: A security group used to allow Fargate containers to receive traffic
    Value: !Ref 'ContainerSecurityGroup'
    Export:
      Name:
        Fn::Sub: "${AWS::StackName}-ContainerSecurityGroup"
```

Contoh 2: file CloudFormation IAc

```
AWSTemplateFormatVersion: '2010-09-09'
Description: Deploy a service on AWS Fargate, hosted in a public subnet, and accessible
 via a public load balancer.
Parameters:
    ContainerPortInput:  # input parameter
        Description: The port to route traffic to
        Type: Number
        Default: 80
    TaskCountInput:      # input parameter
        Description: The default number of Fargate tasks you want running
        Type: Number
        Default: 1
    TaskSizeInput:       # input parameter
        Description: The size of the task you want to run
        Type: String
        Default: x-small
    ContainerImageInput: # input parameter
        Description: The name/url of the container image
        Type: String
        Default: "public.ecr.aws/z9d2n7e1/nginx:1.19.5"
    TaskNameInput:       # input parameter
        Description: Name for your task
        Type: String
```

```
        Default: "my-fargate-instance"
    StackName:           # input parameter
        Description: Name of the environment stack to deploy to
        Type: String
        Default: "my-fargate-environment"
Mappings:
  TaskSizeMap:
    x-small:
      cpu: 256
      memory: 512
    small:
      cpu: 512
      memory: 1024
    medium:
      cpu: 1024
      memory: 2048
    large:
      cpu: 2048
      memory: 4096
    x-large:
      cpu: 4096
      memory: 8192
Resources:
  # A log group for storing the stdout logs from this service's containers
  LogGroup:
    Type: AWS::Logs::LogGroup
    Properties:
      LogGroupName:
        Ref: 'TaskNameInput' # input parameter

  # The task definition. This is a simple metadata description of what
  # container to run, and what resource requirements it has.
  TaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      Family: !Ref 'TaskNameInput'
      Cpu: !FindInMap [TaskSizeMap, !Ref 'TaskSizeInput', cpu]
      Memory: !FindInMap [TaskSizeMap, !Ref 'TaskSizeInput', memory]
      NetworkMode: awsvpc
      RequiresCompatibilities:
        - FARGATE
      ExecutionRoleArn:
        Fn::ImportValue:
```

```
          !Sub "${StackName}-ECSTaskExecutionRole"    # output parameter from another
CloudFormation template
             awslogs-region: !Ref 'AWS::Region'
             awslogs-stream-prefix: !Ref 'TaskNameInput'


 # The service_instance. The service is a resource which allows you to run multiple
 # copies of a type of task, and gather up their logs and metrics, as well
 # as monitor the number of running tasks and replace any that have crashed
 Service:
   Type: AWS::ECS::Service
   DependsOn: LoadBalancerRule
   Properties:
     ServiceName: !Ref 'TaskNameInput'
     Cluster:
       Fn::ImportValue:
         !Sub "${StackName}-ECSCluster"  # output parameter from another
CloudFormation template
     LaunchType: FARGATE
     DeploymentConfiguration:
       MaximumPercent: 200
       MinimumHealthyPercent: 75
     DesiredCount: !Ref 'TaskCountInput'
     NetworkConfiguration:
       AwsvpcConfiguration:
         AssignPublicIp: ENABLED
         SecurityGroups:
           - Fn::ImportValue:
               !Sub "${StackName}-ContainerSecurityGroup"    # output parameter from
another CloudFormation template
         Subnets:
           - Fn::ImportValue:r CloudFormation template
     TaskRoleArn: !Ref "AWS::NoValue"
     ContainerDefinitions:
       - Name: !Ref 'TaskNameInput'
         Cpu: !FindInMap [TaskSizeMap, !Ref 'TaskSizeInput', cpu]
         Memory: !FindInMap [TaskSizeMap, !Ref 'TaskSizeInput', memory]
         Image: !Ref 'ContainerImageInput'             # input parameter
         PortMappings:
           - ContainerPort: !Ref 'ContainerPortInput'  # input parameter

         LogConfiguration:
           LogDriver: 'awslogs'
           Options:
```

```
              awslogs-group: !Ref 'TaskNameInput'
                !Sub "${StackName}-PublicSubnetOne"    # output parameter from another
CloudFormation template
           - Fn::ImportValue:
                !Sub "${StackName}-PublicSubnetTwo"    # output parameter from another
CloudFormation template
     TaskDefinition: !Ref 'TaskDefinition'
     LoadBalancers:
       - ContainerName: !Ref 'TaskNameInput'
         ContainerPort: !Ref 'ContainerPortInput'  # input parameter
         TargetGroupArn: !Ref 'TargetGroup'

 # A target group. This is used for keeping track of all the tasks, and
 # what IP addresses / port numbers they have. You can query it yourself,
 # to use the addresses yourself, but most often this target group is just
 # connected to an application load balancer, or network load balancer, so
 # it can automatically distribute traffic across all the targets.
 TargetGroup:
   Type: AWS::ElasticLoadBalancingV2::TargetGroup
   Properties:
     HealthCheckIntervalSeconds: 6
     HealthCheckPath: /
     HealthCheckProtocol: HTTP
     HealthCheckTimeoutSeconds: 5
     HealthyThresholdCount: 2
     TargetType: ip
     Name: !Ref 'TaskNameInput'
     Port: !Ref 'ContainerPortInput'
     Protocol: HTTP
     UnhealthyThresholdCount: 2
     VpcId:
       Fn::ImportValue:
         !Sub "${StackName}-VPC"    # output parameter from another CloudFormation
template

 # Create a rule on the load balancer for routing traffic to the target group
 LoadBalancerRule:
   Type: AWS::ElasticLoadBalancingV2::ListenerRule
   Properties:
     Actions:
       - TargetGroupArn: !Ref 'TargetGroup'
         Type: 'forward'
     Conditions:
       - Field: path-pattern
```

```
          Values:
            - '*'
      ListenerArn: !Ref PublicLoadBalancerListener
      Priority: 1

  # Enable autoscaling for this service
  ScalableTarget:
    Type: AWS::ApplicationAutoScaling::ScalableTarget
    DependsOn: Service
    Properties:
      ServiceNamespace: 'ecs'
      ScalableDimension: 'ecs:service:DesiredCount'
      ResourceId:
        Fn::Join:
          - '/'
          - - service
            - Fn::ImportValue:
                !Sub "${StackName}-ECSCluster"
            - !Ref 'TaskNameInput'
      MinCapacity: 1
      MaxCapacity: 10
      RoleARN: !Sub arn:aws:iam::${AWS::AccountId}:role/
aws-service-role/ecs.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ECSService

  # Create scaling policies for the service
  ScaleDownPolicy:
    Type: AWS::ApplicationAutoScaling::ScalingPolicy
    DependsOn: ScalableTarget
    Properties:
      PolicyName:
        Fn::Join:
          - '/'
          - - scale
            - !Ref 'TaskNameInput'
            - down
      PolicyType: StepScaling
      ResourceId:
        Fn::Join:
          - '/'
          - - service
            - Fn::ImportValue:
                !Sub "${StackName}-ECSCluster"  # output parameter from another
 CloudFormation template
```

```
            - !Ref 'TaskNameInput'
      ScalableDimension: 'ecs:service:DesiredCount'
      ServiceNamespace: 'ecs'
      StepScalingPolicyConfiguration:
        AdjustmentType: 'ChangeInCapacity'
        StepAdjustments:
          - MetricIntervalUpperBound: 0
            ScalingAdjustment: -1
        MetricAggregationType: 'Average'
        Cooldown: 60

  ScaleUpPolicy:
    Type: AWS::ApplicationAutoScaling::ScalingPolicy
    DependsOn: ScalableTarget
    Properties:
      PolicyName:
        Fn::Join:
          - '/'
          - - scale
            - !Ref 'TaskNameInput'
            - up
      PolicyType: StepScaling
      ResourceId:
        Fn::Join:
          - '/'
          - - service
            - Fn::ImportValue:
                !Sub "${StackName}-ECSCluster"
            - !Ref 'TaskNameInput'
      ScalableDimension: 'ecs:service:DesiredCount'
      ServiceNamespace: 'ecs'
      StepScalingPolicyConfiguration:
        AdjustmentType: 'ChangeInCapacity'
        StepAdjustments:
          - MetricIntervalLowerBound: 0
            MetricIntervalUpperBound: 15
            ScalingAdjustment: 1
          - MetricIntervalLowerBound: 15
            MetricIntervalUpperBound: 25
            ScalingAdjustment: 2
          - MetricIntervalLowerBound: 25
            ScalingAdjustment: 3
        MetricAggregationType: 'Average'
        Cooldown: 60
```

```
# Create alarms to trigger these policies
LowCpuUsageAlarm:
  Type: AWS::CloudWatch::Alarm
  Properties:
    AlarmName:
      Fn::Join:
        - '-'
        - - low-cpu
          - !Ref 'TaskNameInput'
    AlarmDescription:
      Fn::Join:
        - ' '
        - - "Low CPU utilization for service"
          - !Ref 'TaskNameInput'
    MetricName: CPUUtilization
    Namespace: AWS/ECS
    Dimensions:
      - Name: ServiceName
        Value: !Ref 'TaskNameInput'
      - Name: ClusterName
        Value:
          Fn::ImportValue:
            !Sub "${StackName}-ECSCluster"
    Statistic: Average
    Period: 60
    EvaluationPeriods: 1
    Threshold: 20
    ComparisonOperator: LessThanOrEqualToThreshold
    AlarmActions:
      - !Ref ScaleDownPolicy

HighCpuUsageAlarm:
  Type: AWS::CloudWatch::Alarm
  Properties:
    AlarmName:
      Fn::Join:
        - '-'
        - - high-cpu
          - !Ref 'TaskNameInput'
    AlarmDescription:
      Fn::Join:
        - ' '
        - - "High CPU utilization for service"
```

```
            - !Ref 'TaskNameInput'
      MetricName: CPUUtilization
      Namespace: AWS/ECS
      Dimensions:
        - Name: ServiceName
          Value: !Ref 'TaskNameInput'
        - Name: ClusterName
          Value:
            Fn::ImportValue:
              !Sub "${StackName}-ECSCluster"
      Statistic: Average
      Period: 60
      EvaluationPeriods: 1
      Threshold: 70
      ComparisonOperator: GreaterThanOrEqualToThreshold
      AlarmActions:
        - !Ref ScaleUpPolicy

  EcsSecurityGroupIngressFromPublicALB:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      Description: Ingress from the public ALB
      GroupId:
        Fn::ImportValue:
          !Sub "${StackName}-ContainerSecurityGroup"
      IpProtocol: -1
      SourceSecurityGroupId: !Ref 'PublicLoadBalancerSG'

  # Public load balancer, hosted in public subnets that is accessible
  # to the public, and is intended to route traffic to one or more public
  # facing services. This is used for accepting traffic from the public
  # internet and directing it to public facing microservices
  PublicLoadBalancerSG:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Access to the public facing load balancer
      VpcId:
        Fn::ImportValue:
          !Sub "${StackName}-VPC"
      SecurityGroupIngress:
          # Allow access to ALB from anywhere on the internet
          - CidrIp: 0.0.0.0/0
            IpProtocol: -1
```

```
  PublicLoadBalancer:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Scheme: internet-facing
      LoadBalancerAttributes:
      - Key: idle_timeout.timeout_seconds
        Value: '30'
      Subnets:
        # The load balancer is placed into the public subnets, so that traffic
        # from the internet can reach the load balancer directly via the internet
 gateway
        - Fn::ImportValue:
            !Sub "${StackName}-PublicSubnetOne"
        - Fn::ImportValue:
            !Sub "${StackName}-PublicSubnetTwo"
      SecurityGroups: [!Ref 'PublicLoadBalancerSG']

  PublicLoadBalancerListener:
    Type: AWS::ElasticLoadBalancingV2::Listener
    DependsOn:
      - PublicLoadBalancer
    Properties:
      DefaultActions:
        - TargetGroupArn: !Ref 'TargetGroup'
          Type: 'forward'
      LoadBalancerArn: !Ref 'PublicLoadBalancer'
      Port: 80
      Protocol: HTTP
# These output values will be available to other templates to use.
Outputs:
  ServiceEndpoint:        # output
    Description: The URL to access the service
    Value: !Sub "http://${PublicLoadBalancer.DNSName}"
```

Anda dapat menyesuaikan file-file ini untuk digunakan dengan AWS Proton.

Bawa infrastruktur Anda sebagai kode AWS Proton

Dengan sedikit modifikasi, Anda dapat menggunakan Contoh 1 sebagai file infrastruktur sebagai kode (IAc) untuk bundel template lingkungan yang AWS Proton digunakan untuk menyebarkan lingkungan (seperti yang ditunjukkan pada Contoh 3).

Alih-alih menggunakan CloudFormation parameter, Anda menggunakan sintaks Jinja untuk mereferensikan parameter yang telah Anda tentukan dalam file skema berbasis API Terbuka. Parameter input ini dikomentari untuk kenyamanan Anda dan direferensikan beberapa kali dalam file IAC. Dengan cara ini, AWS Proton dapat mengaudit dan memeriksa nilai parameter. Itu juga dapat mencocokkan dan menyisipkan nilai parameter output dalam satu file IAc ke parameter dalam file IAc lain.

Sebagai administrator, Anda dapat menambahkan AWS Proton `environment.inputs.` namespace ke parameter input. Saat Anda mereferensikan output file IAc lingkungan dalam file iAc layanan, Anda dapat menambahkan `environment.outputs.` namespace ke output (misalnya,). `environment.outputs.ClusterName` Terakhir, Anda mengelilinginya dengan kawat gigi keriting dan tanda kutip.

Dengan modifikasi ini, file CloudFormation IAC Anda dapat digunakan oleh AWS Proton.

Contoh 3: infrastruktur AWS Proton lingkungan sebagai file kode

```
AWSTemplateFormatVersion: '2010-09-09'
Description: AWS Fargate cluster running containers in a public subnet. Only supports
             public facing load balancer, and public service discovery prefixes.
Mappings:
  # The VPC and subnet configuration is passed in via the environment spec.
  SubnetConfig:
    VPC:
      CIDR: '{{ environment.inputs.vpc_cidr}}'        # input parameter
    PublicOne:
      CIDR: '{{ environment.inputs.subnet_one_cidr}}' # input parameter
    PublicTwo:
      CIDR: '{{ environment.inputs.subnet_two_cidr}}' # input parameter
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      EnableDnsSupport: true
      EnableDnsHostnames: true
      CidrBlock: !FindInMap ['SubnetConfig', 'VPC', 'CIDR']

  # Two public subnets, where containers will have public IP addresses
  PublicSubnetOne:
    Type: AWS::EC2::Subnet
    Properties:
      AvailabilityZone:
```

```
        Fn::Select:
        - 0
        - Fn::GetAZs: {Ref: 'AWS::Region'}
      VpcId: !Ref 'VPC'
      CidrBlock: !FindInMap ['SubnetConfig', 'PublicOne', 'CIDR']
      MapPublicIpOnLaunch: true

  PublicSubnetTwo:
    Type: AWS::EC2::Subnet
    Properties:
      AvailabilityZone:
        Fn::Select:
        - 1
        - Fn::GetAZs: {Ref: 'AWS::Region'}
      VpcId: !Ref 'VPC'
      CidrBlock: !FindInMap ['SubnetConfig', 'PublicTwo', 'CIDR']
      MapPublicIpOnLaunch: true

  # Setup networking resources for the public subnets. Containers
  # in the public subnets have public IP addresses and the routing table
  # sends network traffic via the internet gateway.
  InternetGateway:
    Type: AWS::EC2::InternetGateway
  GatewayAttachement:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      VpcId: !Ref 'VPC'
      InternetGatewayId: !Ref 'InternetGateway'
  PublicRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref 'VPC'
  PublicRoute:
    Type: AWS::EC2::Route
    DependsOn: GatewayAttachement
    Properties:
      RouteTableId: !Ref 'PublicRouteTable'
      DestinationCidrBlock: '0.0.0.0/0'
      GatewayId: !Ref 'InternetGateway'
  PublicSubnetOneRouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref PublicSubnetOne
      RouteTableId: !Ref PublicRouteTable
```

```
  PublicSubnetTwoRouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref PublicSubnetTwo
      RouteTableId: !Ref PublicRouteTable

  # ECS Resources
  ECSCluster:
    Type: AWS::ECS::Cluster

  # A security group for the containers we will run in Fargate.
  # Rules are added to this security group based on what ingress you
  # add for the cluster.
  ContainerSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Access to the Fargate containers
      VpcId: !Ref 'VPC'

  # This is a role which is used by the ECS tasks themselves.
  ECSTaskExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
        - Effect: Allow
          Principal:
            Service: [ecs-tasks.amazonaws.com]
          Action: ['sts:AssumeRole']
      Path: /
      ManagedPolicyArns:
        - 'arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy'

# These output values are available to service infrastructure as code files as outputs,
 when given the
# the 'service_instance.environment.outputs.' namespace, for example,
 service_instance.environment.outputs.ClusterName.

Outputs:
  ClusterName:                                          # output
    Description: The name of the ECS cluster
    Value: !Ref 'ECSCluster'
  ECSTaskExecutionRole:                                 # output
    Description: The ARN of the ECS role
```

```
    Value: !GetAtt 'ECSTaskExecutionRole.Arn'
  VpcId:                                                  # output
    Description: The ID of the VPC that this stack is deployed in
    Value: !Ref 'VPC'
  PublicSubnetOne:                                        # output
    Description: Public subnet one
    Value: !Ref 'PublicSubnetOne'
  PublicSubnetTwo:                                        # output
    Description: Public subnet two
    Value: !Ref 'PublicSubnetTwo'
  ContainerSecurityGroup:                                 # output
    Description: A security group used to allow Fargate containers to receive traffic
    Value: !Ref 'ContainerSecurityGroup'
```

File IAc di Contoh 1 dan Contoh 3 menghasilkan CloudFormation tumpukan yang sedikit berbeda. Parameter ditampilkan secara berbeda dalam file template stack. File template CloudFormation tumpukan Contoh 1 menampilkan label parameter (kunci) dalam tampilan template tumpukan. File template tumpukan AWS Proton CloudFormation infrastruktur Contoh 3 menampilkan nilai parameter. AWS Proton parameter input tidak muncul di tampilan parameter CloudFormation tumpukan konsol.

Dalam Contoh 4, file iAc AWS Proton layanan sesuai dengan Contoh 2.

Contoh 4: file iAc contoh AWS Proton layanan

```
AWSTemplateFormatVersion: '2010-09-09'
Description: Deploy a service on AWS Fargate, hosted in a public subnet, and accessible
 via a public load balancer.
Mappings:
  TaskSize:
    x-small:
      cpu: 256
      memory: 512
    small:
      cpu: 512
      memory: 1024
    medium:
      cpu: 1024
      memory: 2048
    large:
      cpu: 2048
      memory: 4096
    x-large:
      cpu: 4096
```

```
      memory: 8192
Resources:
  # A log group for storing the stdout logs from this service's containers
  LogGroup:
    Type: AWS::Logs::LogGroup
    Properties:
      LogGroupName: '{{service_instance.name}}' # resource parameter

  # The task definition. This is a simple metadata description of what
  # container to run, and what resource requirements it has.
  TaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      Family: '{{service_instance.name}}'
      Cpu: !FindInMap [TaskSize, {{service_instance.inputs.task_size}}, cpu] # input
 parameter
      Memory: !FindInMap [TaskSize, {{service_instance.inputs.task_size}}, memory]
      NetworkMode: awsvpc
      RequiresCompatibilities:
        - FARGATE
      ExecutionRoleArn: '{{environment.outputs.ECSTaskExecutionRole}}' # output from an
 environment infrastructure as code file
      TaskRoleArn: !Ref "AWS::NoValue"
      ContainerDefinitions:
        - Name: '{{service_instance.name}}'
          Cpu: !FindInMap [TaskSize, {{service_instance.inputs.task_size}}, cpu]
          Memory: !FindInMap [TaskSize, {{service_instance.inputs.task_size}}, memory]
          Image: '{{service_instance.inputs.image}}'
          PortMappings:
            - ContainerPort: '{{service_instance.inputs.port}}' # input parameter
          LogConfiguration:
            LogDriver: 'awslogs'
            Options:
              awslogs-group: '{{service_instance.name}}'
              awslogs-region: !Ref 'AWS::Region'
              awslogs-stream-prefix: '{{service_instance.name}}'

  # The service_instance. The service is a resource which allows you to run multiple
  # copies of a type of task, and gather up their logs and metrics, as well
  # as monitor the number of running tasks and replace any that have crashed
  Service:
    Type: AWS::ECS::Service
    DependsOn: LoadBalancerRule
    Properties:
```

```
      ServiceName: '{{service_instance.name}}'
      Cluster: '{{environment.outputs.ClusterName}}' # output from an environment
infrastructure as code file
      LaunchType: FARGATE
      DeploymentConfiguration:
        MaximumPercent: 200
        MinimumHealthyPercent: 75
      DesiredCount: '{{service_instance.inputs.desired_count}}'       # input parameter
      NetworkConfiguration:
        AwsvpcConfiguration:
          AssignPublicIp: ENABLED
          SecurityGroups:
            - '{{environment.outputs.ContainerSecurityGroup}}' # output from an
environment infrastructure as code file
          Subnets:
            - '{{environment.outputs.PublicSubnetOne}}'        # output from an
environment infrastructure as code file
            - '{{environment.outputs.PublicSubnetTwo}}'
      TaskDefinition: !Ref 'TaskDefinition'
      LoadBalancers:
        - ContainerName: '{{service_instance.name}}'
          ContainerPort: '{{service_instance.inputs.port}}'
          TargetGroupArn: !Ref 'TargetGroup'

  # A target group. This is used for keeping track of all the tasks, and
  # what IP addresses / port numbers they have. You can query it yourself,
  # to use the addresses yourself, but most often this target group is just
  # connected to an application load balancer, or network load balancer, so
  # it can automatically distribute traffic across all the targets.
  TargetGroup:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
      HealthCheckIntervalSeconds: 6
      HealthCheckPath: /
      HealthCheckProtocol: HTTP
      HealthCheckTimeoutSeconds: 5
      HealthyThresholdCount: 2
      TargetType: ip
      Name: '{{service_instance.name}}'
      Port: '{{service_instance.inputs.port}}'
      Protocol: HTTP
      UnhealthyThresholdCount: 2
      VpcId: '{{environment.outputs.VpcId}}' # output from an environment
infrastructure as code file
```

```
  # Create a rule on the load balancer for routing traffic to the target group
  LoadBalancerRule:
    Type: AWS::ElasticLoadBalancingV2::ListenerRule
    Properties:
      Actions:
        - TargetGroupArn: !Ref 'TargetGroup'
          Type: 'forward'
      Conditions:
        - Field: path-pattern
          Values:
            - '*'
      ListenerArn: !Ref PublicLoadBalancerListener
      Priority: 1

  # Enable autoscaling for this service
  ScalableTarget:
    Type: AWS::ApplicationAutoScaling::ScalableTarget
    DependsOn: Service
    Properties:
      ServiceNamespace: 'ecs'
      ScalableDimension: 'ecs:service:DesiredCount'
      ResourceId:
        Fn::Join:
          - '/'
          - - service
            - '{{environment.outputs.ClusterName}}' # output from an environment
 infrastructure as code file
            - '{{service_instance.name}}'
      MinCapacity: 1
      MaxCapacity: 10
      RoleARN: !Sub arn:aws:iam::${AWS::AccountId}:role/
aws-service-role/ecs.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ECSService

  # Create scaling policies for the service
  ScaleDownPolicy:
    Type: AWS::ApplicationAutoScaling::ScalingPolicy
    DependsOn: ScalableTarget
    Properties:
      PolicyName:
        Fn::Join:
          - '/'
          - - scale
```

```
            - '{{service_instance.name}}'
            - down
      PolicyType: StepScaling
      ResourceId:
        Fn::Join:
          - '/'
          - - service
            - '{{environment.outputs.ClusterName}}'
            - '{{service_instance.name}}'
      ScalableDimension: 'ecs:service:DesiredCount'
      ServiceNamespace: 'ecs'
      StepScalingPolicyConfiguration:
        AdjustmentType: 'ChangeInCapacity'
        StepAdjustments:
          - MetricIntervalUpperBound: 0
            ScalingAdjustment: -1
        MetricAggregationType: 'Average'
        Cooldown: 60

  ScaleUpPolicy:
    Type: AWS::ApplicationAutoScaling::ScalingPolicy
    DependsOn: ScalableTarget
    Properties:
      PolicyName:
        Fn::Join:
          - '/'
          - - scale
            - '{{service_instance.name}}'
            - up
      PolicyType: StepScaling
      ResourceId:
        Fn::Join:
          - '/'
          - - service
            - '{{environment.outputs.ClusterName}}'
            - '{{service_instance.name}}'
      ScalableDimension: 'ecs:service:DesiredCount'
      ServiceNamespace: 'ecs'
      StepScalingPolicyConfiguration:
        AdjustmentType: 'ChangeInCapacity'
        StepAdjustments:
          - MetricIntervalLowerBound: 0
            MetricIntervalUpperBound: 15
            ScalingAdjustment: 1
```

```
        - MetricIntervalLowerBound: 15
          MetricIntervalUpperBound: 25
          ScalingAdjustment: 2
        - MetricIntervalLowerBound: 25
          ScalingAdjustment: 3
      MetricAggregationType: 'Average'
      Cooldown: 60

# Create alarms to trigger these policies
LowCpuUsageAlarm:
  Type: AWS::CloudWatch::Alarm
  Properties:
    AlarmName:
      Fn::Join:
        - '-'
        - - low-cpu
          - '{{service_instance.name}}'
    AlarmDescription:
      Fn::Join:
        - ' '
        - - "Low CPU utilization for service"
          - '{{service_instance.name}}'
    MetricName: CPUUtilization
    Namespace: AWS/ECS
    Dimensions:
      - Name: ServiceName
        Value: '{{service_instance.name}}'
      - Name: ClusterName
        Value:
          '{{environment.outputs.ClusterName}}'
    Statistic: Average
    Period: 60
    EvaluationPeriods: 1
    Threshold: 20
    ComparisonOperator: LessThanOrEqualToThreshold
    AlarmActions:
      - !Ref ScaleDownPolicy

HighCpuUsageAlarm:
  Type: AWS::CloudWatch::Alarm
  Properties:
    AlarmName:
      Fn::Join:
        - '-'
```

```
          - - high-cpu
            - '{{service_instance.name}}'
      AlarmDescription:
        Fn::Join:
          - ' '
          - - "High CPU utilization for service"
            - '{{service_instance.name}}'
      MetricName: CPUUtilization
      Namespace: AWS/ECS
      Dimensions:
        - Name: ServiceName
          Value: '{{service_instance.name}}'
        - Name: ClusterName
          Value:
            '{{environment.outputs.ClusterName}}'
      Statistic: Average
      Period: 60
      EvaluationPeriods: 1
      Threshold: 70
      ComparisonOperator: GreaterThanOrEqualToThreshold
      AlarmActions:
        - !Ref ScaleUpPolicy

  EcsSecurityGroupIngressFromPublicALB:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      Description: Ingress from the public ALB
      GroupId: '{{environment.outputs.ContainerSecurityGroup}}'
      IpProtocol: -1
      SourceSecurityGroupId: !Ref 'PublicLoadBalancerSG'

  # Public load balancer, hosted in public subnets that is accessible
  # to the public, and is intended to route traffic to one or more public
  # facing services. This is used for accepting traffic from the public
  # internet and directing it to public facing microservices
  PublicLoadBalancerSG:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Access to the public facing load balancer
      VpcId: '{{environment.outputs.VpcId}}'
      SecurityGroupIngress:
          # Allow access to ALB from anywhere on the internet
          - CidrIp: 0.0.0.0/0
            IpProtocol: -1
```

```
  PublicLoadBalancer:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Scheme: internet-facing
      LoadBalancerAttributes:
      - Key: idle_timeout.timeout_seconds
        Value: '30'
      Subnets:
        # The load balancer is placed into the public subnets, so that traffic
        # from the internet can reach the load balancer directly via the internet
 gateway
        - '{{environment.outputs.PublicSubnetOne}}'
        - '{{environment.outputs.PublicSubnetTwo}}'
      SecurityGroups: [!Ref 'PublicLoadBalancerSG']

  PublicLoadBalancerListener:
    Type: AWS::ElasticLoadBalancingV2::Listener
    DependsOn:
      - PublicLoadBalancer
    Properties:
      DefaultActions:
        - TargetGroupArn: !Ref 'TargetGroup'
          Type: 'forward'
      LoadBalancerArn: !Ref 'PublicLoadBalancer'
      Port: 80
      Protocol: HTTP
Outputs:
  ServiceEndpoint:         # output
    Description: The URL to access the service
    Value: !Sub "http://${PublicLoadBalancer.DNSName}"
```

Dalam Contoh 5, file iAc AWS Proton pipeline menyediakan infrastruktur pipeline untuk mendukung instance layanan yang disediakan oleh Contoh 4.

Contoh 5: file iAc pipa AWS Proton layanan

```
Resources:
  ECRRepo:
    Type: AWS::ECR::Repository
    DeletionPolicy: Retain
  BuildProject:
    Type: AWS::CodeBuild::Project
```

```
    Properties:
      Artifacts:
        Type: CODEPIPELINE
      Environment:
        ComputeType: BUILD_GENERAL1_SMALL
        Image: aws/codebuild/amazonlinux2-x86_64-standard:3.0
        PrivilegedMode: true
        Type: LINUX_CONTAINER
        EnvironmentVariables:
        - Name: repo_name
          Type: PLAINTEXT
          Value: !Ref ECRRepo
        - Name: service_name
          Type: PLAINTEXT
          Value: '{{ service.name }}'    # resource parameter
      ServiceRole:
        Fn::GetAtt:
          - PublishRole
          - Arn
      Source:
        BuildSpec:
          Fn::Join:
            - ""
            - - >-
                {
                  "version": "0.2",
                  "phases": {
                    "install": {
                      "runtime-versions": {
                        "docker": 18
                      },
                      "commands": [
                        "pip3 install --upgrade --user awscli",
                        "echo
 'f6bd1536a743ab170b35c94ed4c7c4479763356bd543af5d391122f4af852460  yq_linux_amd64' >
 yq_linux_amd64.sha",
                        "wget https://github.com/mikefarah/yq/releases/download/3.4.0/
yq_linux_amd64",
                        "sha256sum -c yq_linux_amd64.sha",
                        "mv yq_linux_amd64 /usr/bin/yq",
                        "chmod +x /usr/bin/yq"
                      ]
                    },
                    "pre_build": {
```

```
                    "commands": [
                      "cd $CODEBUILD_SRC_DIR",
                      "$(aws ecr get-login --no-include-email --region
 $AWS_DEFAULT_REGION)",
                      "{{ pipeline.inputs.unit_test_command }}",    # input parameter
                    ]
                  },
                  "build": {
                    "commands": [
                      "IMAGE_REPO_NAME=$repo_name",
                      "IMAGE_TAG=$CODEBUILD_BUILD_NUMBER",
                      "IMAGE_ID=
            - Ref: AWS::AccountId
            - >-
              .dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:
$IMAGE_TAG",
                      "docker build -t $IMAGE_REPO_NAME:$IMAGE_TAG -f
 {{ pipeline.inputs.dockerfile }} .",     # input parameter
                      "docker tag $IMAGE_REPO_NAME:$IMAGE_TAG $IMAGE_ID;",
                      "docker push $IMAGE_ID"
                    ]
                  },
                  "post_build": {
                    "commands": [
                      "aws proton --region $AWS_DEFAULT_REGION get-service --name
 $service_name | jq -r .service.spec > service.yaml",
                      "yq w service.yaml 'instances[*].spec.image' \"$IMAGE_ID\" >
 rendered_service.yaml"
                    ]
                  }
                },
                "artifacts": {
                  "files": [
                    "rendered_service.yaml"
                  ]
                }
              }
        Type: CODEPIPELINE
      EncryptionKey:
        Fn::GetAtt:
          - PipelineArtifactsBucketEncryptionKey
          - Arn
{% for service_instance in service_instances %}
  Deploy{{loop.index}}Project:
```

```
    Type: AWS::CodeBuild::Project
    Properties:
      Artifacts:
        Type: CODEPIPELINE
      Environment:
        ComputeType: BUILD_GENERAL1_SMALL
        Image: aws/codebuild/amazonlinux2-x86_64-standard:3.0
        PrivilegedMode: false
        Type: LINUX_CONTAINER
        EnvironmentVariables:
        - Name: service_name
          Type: PLAINTEXT
          Value:  '{{service.name}}'          # resource parameter
        - Name: service_instance_name
          Type: PLAINTEXT
          Value: '{{service_instance.name}}'  # resource parameter
      ServiceRole:
        Fn::GetAtt:
          - DeploymentRole
          - Arn
      Source:
        BuildSpec: >-
          {
            "version": "0.2",
            "phases": {
              "build": {
                "commands": [
                  "pip3 install --upgrade --user awscli",
                  "aws proton --region $AWS_DEFAULT_REGION update-service-instance
 --deployment-type CURRENT_VERSION --name $service_instance_name --service-name
 $service_name --spec file://rendered_service.yaml",
                  "aws proton --region $AWS_DEFAULT_REGION wait service-instance-
deployed --name $service_instance_name --service-name $service_name"
                ]
              }
            }
          }
        Type: CODEPIPELINE
      EncryptionKey:
        Fn::GetAtt:
          - PipelineArtifactsBucketEncryptionKey
          - Arn
{% endfor %}
  # This role is used to build and publish an image to ECR
```

```
  PublishRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              Service: codebuild.amazonaws.com
        Version: "2012-10-17"
  PublishRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - logs:CreateLogGroup
              - logs:CreateLogStream
              - logs:PutLogEvents
            Effect: Allow
            Resource:
              - Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":logs:"
                    - Ref: AWS::Region
                    - ":"
                    - Ref: AWS::AccountId
                    - :log-group:/aws/codebuild/
                    - Ref: BuildProject
              - Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":logs:"
                    - Ref: AWS::Region
                    - ":"
                    - Ref: AWS::AccountId
                    - :log-group:/aws/codebuild/
                    - Ref: BuildProject
                    - :*
          - Action:
              - codebuild:CreateReportGroup
```

```
              - codebuild:CreateReport
              - codebuild:UpdateReport
              - codebuild:BatchPutTestCases
            Effect: Allow
            Resource:
              Fn::Join:
                - ""
                - - "arn:"
                  - Ref: AWS::Partition
                  - ":codebuild:"
                  - Ref: AWS::Region
                  - ":"
                  - Ref: AWS::AccountId
                  - :report-group/
                  - Ref: BuildProject
                  - -*
          - Action:
              - ecr:GetAuthorizationToken
            Effect: Allow
            Resource: "*"
          - Action:
              - ecr:BatchCheckLayerAvailability
              - ecr:CompleteLayerUpload
              - ecr:GetAuthorizationToken
              - ecr:InitiateLayerUpload
              - ecr:PutImage
              - ecr:UploadLayerPart
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - ECRRepo
                - Arn
          - Action:
              - proton:GetService
            Effect: Allow
            Resource: "*"
          - Action:
              - s3:GetObject*
              - s3:GetBucket*
              - s3:List*
              - s3:DeleteObject*
              - s3:PutObject*
              - s3:Abort*
            Effect: Allow
```

```
            Resource:
              - Fn::GetAtt:
                  - PipelineArtifactsBucket
                  - Arn
              - Fn::Join:
                  - ""
                  - - Fn::GetAtt:
                        - PipelineArtifactsBucket
                        - Arn
                    - /*
          - Action:
              - kms:Decrypt
              - kms:DescribeKey
              - kms:Encrypt
              - kms:ReEncrypt*
              - kms:GenerateDataKey*
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineArtifactsBucketEncryptionKey
                - Arn
          - Action:
              - kms:Decrypt
              - kms:Encrypt
              - kms:ReEncrypt*
              - kms:GenerateDataKey*
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineArtifactsBucketEncryptionKey
                - Arn
        Version: "2012-10-17"
      PolicyName: PublishRoleDefaultPolicy
      Roles:
        - Ref: PublishRole

  DeploymentRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
```

```
            Service: codebuild.amazonaws.com
      Version: "2012-10-17"
  DeploymentRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - logs:CreateLogGroup
              - logs:CreateLogStream
              - logs:PutLogEvents
            Effect: Allow
            Resource:
              - Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":logs:"
                    - Ref: AWS::Region
                    - ":"
                    - Ref: AWS::AccountId
                    - :log-group:/aws/codebuild/Deploy*Project*
              - Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":logs:"
                    - Ref: AWS::Region
                    - ":"
                    - Ref: AWS::AccountId
                    - :log-group:/aws/codebuild/Deploy*Project:*
          - Action:
              - codebuild:CreateReportGroup
              - codebuild:CreateReport
              - codebuild:UpdateReport
              - codebuild:BatchPutTestCases
            Effect: Allow
            Resource:
              Fn::Join:
                - ""
                - - "arn:"
                  - Ref: AWS::Partition
                  - ":codebuild:"
                  - Ref: AWS::Region
```

```
                - ":"
                - Ref: AWS::AccountId
                - :report-group/Deploy*Project
                - -*
          - Action:
              - proton:UpdateServiceInstance
              - proton:GetServiceInstance
            Effect: Allow
            Resource: "*"
          - Action:
              - s3:GetObject*
              - s3:GetBucket*
              - s3:List*
            Effect: Allow
            Resource:
              - Fn::GetAtt:
                  - PipelineArtifactsBucket
                  - Arn
              - Fn::Join:
                  - ""
                  - - Fn::GetAtt:
                        - PipelineArtifactsBucket
                        - Arn
                    - /*
          - Action:
              - kms:Decrypt
              - kms:DescribeKey
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineArtifactsBucketEncryptionKey
                - Arn
          - Action:
              - kms:Decrypt
              - kms:Encrypt
              - kms:ReEncrypt*
              - kms:GenerateDataKey*
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineArtifactsBucketEncryptionKey
                - Arn
        Version: "2012-10-17"
      PolicyName: DeploymentRoleDefaultPolicy
```

```
      Roles:
        - Ref: DeploymentRole
  PipelineArtifactsBucketEncryptionKey:
    Type: AWS::KMS::Key
    Properties:
      KeyPolicy:
        Statement:
          - Action:
              - kms:Create*
              - kms:Describe*
              - kms:Enable*
              - kms:List*
              - kms:Put*
              - kms:Update*
              - kms:Revoke*
              - kms:Disable*
              - kms:Get*
              - kms:Delete*
              - kms:ScheduleKeyDeletion
              - kms:CancelKeyDeletion
              - kms:GenerateDataKey
              - kms:TagResource
              - kms:UntagResource
            Effect: Allow
            Principal:
              AWS:
                Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":iam::"
                    - Ref: AWS::AccountId
                    - :root
            Resource: "*"
          - Action:
              - kms:Decrypt
              - kms:DescribeKey
              - kms:Encrypt
              - kms:ReEncrypt*
              - kms:GenerateDataKey*
            Effect: Allow
            Principal:
              AWS:
                Fn::GetAtt:
```

```
            - PipelineRole
            - Arn
      Resource: "*"
    - Action:
        - kms:Decrypt
        - kms:DescribeKey
        - kms:Encrypt
        - kms:ReEncrypt*
        - kms:GenerateDataKey*
      Effect: Allow
      Principal:
        AWS:
          Fn::GetAtt:
            - PublishRole
            - Arn
      Resource: "*"
    - Action:
        - kms:Decrypt
        - kms:Encrypt
        - kms:ReEncrypt*
        - kms:GenerateDataKey*
      Effect: Allow
      Principal:
        AWS:
          Fn::GetAtt:
            - PublishRole
            - Arn
      Resource: "*"
    - Action:
        - kms:Decrypt
        - kms:DescribeKey
      Effect: Allow
      Principal:
        AWS:
          Fn::GetAtt:
            - DeploymentRole
            - Arn
      Resource: "*"
    - Action:
        - kms:Decrypt
        - kms:Encrypt
        - kms:ReEncrypt*
        - kms:GenerateDataKey*
      Effect: Allow
```

```
            Principal:
              AWS:
                Fn::GetAtt:
                  - DeploymentRole
                  - Arn
            Resource: "*"
        Version: "2012-10-17"
    UpdateReplacePolicy: Delete
    DeletionPolicy: Delete
  PipelineArtifactsBucket:
    Type: AWS::S3::Bucket
    Properties:
      VersioningConfiguration:
        Status: Enabled
      BucketEncryption:
        ServerSideEncryptionConfiguration:
          - ServerSideEncryptionByDefault:
              KMSMasterKeyID:
                Fn::GetAtt:
                  - PipelineArtifactsBucketEncryptionKey
                  - Arn
              SSEAlgorithm: aws:kms
      PublicAccessBlockConfiguration:
        BlockPublicAcls: true
        BlockPublicPolicy: true
        IgnorePublicAcls: true
        RestrictPublicBuckets: true
    UpdateReplacePolicy: Retain
    DeletionPolicy: Retain
  PipelineArtifactsBucketEncryptionKeyAlias:
    Type: AWS::KMS::Alias
    Properties:
      AliasName: 'alias/codepipeline-encryption-key-{{ service.name }}'
      TargetKeyId:
        Fn::GetAtt:
          - PipelineArtifactsBucketEncryptionKey
          - Arn
    UpdateReplacePolicy: Delete
    DeletionPolicy: Delete
  PipelineRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
```

```
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              Service: codepipeline.amazonaws.com
        Version: "2012-10-17"
  PipelineRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - s3:GetObject*
              - s3:GetBucket*
              - s3:List*
              - s3:DeleteObject*
              - s3:PutObject*
              - s3:Abort*
            Effect: Allow
            Resource:
              - Fn::GetAtt:
                  - PipelineArtifactsBucket
                  - Arn
              - Fn::Join:
                  - ""
                  - - Fn::GetAtt:
                        - PipelineArtifactsBucket
                        - Arn
                    - /*
          - Action:
              - kms:Decrypt
              - kms:DescribeKey
              - kms:Encrypt
              - kms:ReEncrypt*
              - kms:GenerateDataKey*
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineArtifactsBucketEncryptionKey
                - Arn
          - Action: codestar-connections:*
            Effect: Allow
            Resource: "*"
          - Action: sts:AssumeRole
            Effect: Allow
```

```
            Resource:
              Fn::GetAtt:
                - PipelineBuildCodePipelineActionRole
                - Arn
          - Action: sts:AssumeRole
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineDeployCodePipelineActionRole
                - Arn
        Version: "2012-10-17"
      PolicyName: PipelineRoleDefaultPolicy
      Roles:
        - Ref: PipelineRole
  Pipeline:
    Type: AWS::CodePipeline::Pipeline
    Properties:
      RoleArn:
        Fn::GetAtt:
          - PipelineRole
          - Arn
      Stages:
        - Actions:
            - ActionTypeId:
                Category: Source
                Owner: AWS
                Provider: CodeStarSourceConnection
                Version: "1"
              Configuration:
                ConnectionArn: '{{ service.repository_connection_arn }}'
                FullRepositoryId: '{{ service.repository_id }}'
                BranchName: '{{ service.branch_name }}'
              Name: Checkout
              OutputArtifacts:
                - Name: Artifact_Source_Checkout
              RunOrder: 1
          Name: Source
        - Actions:
            - ActionTypeId:
                Category: Build
                Owner: AWS
                Provider: CodeBuild
                Version: "1"
              Configuration:
```

```
                ProjectName:
                  Ref: BuildProject
              InputArtifacts:
                - Name: Artifact_Source_Checkout
              Name: Build
              OutputArtifacts:
                - Name: BuildOutput
              RoleArn:
                Fn::GetAtt:
                  - PipelineBuildCodePipelineActionRole
                  - Arn
              RunOrder: 1
          Name: Build {%- for service_instance in service_instances %}
        - Actions:
            - ActionTypeId:
                Category: Build
                Owner: AWS
                Provider: CodeBuild
                Version: "1"
              Configuration:
                ProjectName:
                  Ref: Deploy{{loop.index}}Project
              InputArtifacts:
                - Name: BuildOutput
              Name: Deploy
              RoleArn:
                Fn::GetAtt:
                  - PipelineDeployCodePipelineActionRole
                  - Arn
              RunOrder: 1
          Name: 'Deploy{{service_instance.name}}'
{%- endfor %}
      ArtifactStore:
        EncryptionKey:
          Id:
            Fn::GetAtt:
              - PipelineArtifactsBucketEncryptionKey
              - Arn
          Type: KMS
        Location:
          Ref: PipelineArtifactsBucket
        Type: S3
    DependsOn:
      - PipelineRoleDefaultPolicy
```

```
      - PipelineRole
  PipelineBuildCodePipelineActionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              AWS:
                Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":iam::"
                    - Ref: AWS::AccountId
                    - :root
        Version: "2012-10-17"
  PipelineBuildCodePipelineActionRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - codebuild:BatchGetBuilds
              - codebuild:StartBuild
              - codebuild:StopBuild
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - BuildProject
                - Arn
        Version: "2012-10-17"
      PolicyName: PipelineBuildCodePipelineActionRoleDefaultPolicy
      Roles:
        - Ref: PipelineBuildCodePipelineActionRole
  PipelineDeployCodePipelineActionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
```

```
              AWS:
                Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":iam::"
                    - Ref: AWS::AccountId
                    - :root
        Version: "2012-10-17"
  PipelineDeployCodePipelineActionRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - codebuild:BatchGetBuilds
              - codebuild:StartBuild
              - codebuild:StopBuild
            Effect: Allow
            Resource:
              Fn::Join:
                - ""
                - - "arn:"
                  - Ref: AWS::Partition
                  - ":codebuild:"
                  - Ref: AWS::Region
                  - ":"
                  - Ref: AWS::AccountId
                  - ":project/Deploy*"
        Version: "2012-10-17"
      PolicyName: PipelineDeployCodePipelineActionRoleDefaultPolicy
      Roles:
        - Ref: PipelineDeployCodePipelineActionRole
Outputs:
  PipelineEndpoint:
    Description: The URL to access the pipeline
    Value: !Sub "https://${AWS::Region}.console.aws.amazon.com/codesuite/codepipeline/
pipelines/${Pipeline}/view?region=${AWS::Region}"

                ]
              }
            }
          }
        Type: CODEPIPELINE
```

```
      EncryptionKey:
        Fn::GetAtt:
          - PipelineArtifactsBucketEncryptionKey
          - Arn
{% endfor %}
  # This role is used to build and publish an image to ECR
  PublishRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              Service: codebuild.amazonaws.com
        Version: "2012-10-17"
  PublishRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - logs:CreateLogGroup
              - logs:CreateLogStream
              - logs:PutLogEvents
            Effect: Allow
            Resource:
              - Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":logs:"
                    - Ref: AWS::Region
                    - ":"
                    - Ref: AWS::AccountId
                    - :log-group:/aws/codebuild/
                    - Ref: BuildProject
              - Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":logs:"
                    - Ref: AWS::Region
                    - ":"
```

```
                - Ref: AWS::AccountId
                - :log-group:/aws/codebuild/
                - Ref: BuildProject
                - :*
        - Action:
            - codebuild:CreateReportGroup
            - codebuild:CreateReport
            - codebuild:UpdateReport
            - codebuild:BatchPutTestCases
          Effect: Allow
          Resource:
            Fn::Join:
              - ""
              - - "arn:"
                - Ref: AWS::Partition
                - ":codebuild:"
                - Ref: AWS::Region
                - ":"
                - Ref: AWS::AccountId
                - :report-group/
                - Ref: BuildProject
                - -*
        - Action:
            - ecr:GetAuthorizationToken
          Effect: Allow
          Resource: "*"
        - Action:
            - ecr:BatchCheckLayerAvailability
            - ecr:CompleteLayerUpload
            - ecr:GetAuthorizationToken
            - ecr:InitiateLayerUpload
            - ecr:PutImage
            - ecr:UploadLayerPart
          Effect: Allow
          Resource:
            Fn::GetAtt:
              - ECRRepo
              - Arn
        - Action:
            - proton:GetService
          Effect: Allow
          Resource: "*"
        - Action:
            - s3:GetObject*
```

```
              - s3:GetBucket*
              - s3:List*
              - s3:DeleteObject*
              - s3:PutObject*
              - s3:Abort*
            Effect: Allow
            Resource:
              - Fn::GetAtt:
                  - PipelineArtifactsBucket
                  - Arn
              - Fn::Join:
                  - ""
                  - - Fn::GetAtt:
                        - PipelineArtifactsBucket
                        - Arn
                    - /*
          - Action:
              - kms:Decrypt
              - kms:DescribeKey
              - kms:Encrypt
              - kms:ReEncrypt*
              - kms:GenerateDataKey*
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineArtifactsBucketEncryptionKey
                - Arn
          - Action:
              - kms:Decrypt
              - kms:Encrypt
              - kms:ReEncrypt*
              - kms:GenerateDataKey*
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineArtifactsBucketEncryptionKey
                - Arn
        Version: "2012-10-17"
      PolicyName: PublishRoleDefaultPolicy
      Roles:
        - Ref: PublishRole

  DeploymentRole:
    Type: AWS::IAM::Role
```

```
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              Service: codebuild.amazonaws.com
        Version: "2012-10-17"
  DeploymentRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - logs:CreateLogGroup
              - logs:CreateLogStream
              - logs:PutLogEvents
            Effect: Allow
            Resource:
              - Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":logs:"
                    - Ref: AWS::Region
                    - ":"
                    - Ref: AWS::AccountId
                    - :log-group:/aws/codebuild/Deploy*Project*
              - Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":logs:"
                    - Ref: AWS::Region
                    - ":"
                    - Ref: AWS::AccountId
                    - :log-group:/aws/codebuild/Deploy*Project:*
          - Action:
              - codebuild:CreateReportGroup
              - codebuild:CreateReport
              - codebuild:UpdateReport
              - codebuild:BatchPutTestCases
            Effect: Allow
            Resource:
```

```
            Fn::Join:
              - ""
              - - "arn:"
                - Ref: AWS::Partition
                - ":codebuild:"
                - Ref: AWS::Region
                - ":"
                - Ref: AWS::AccountId
                - :report-group/Deploy*Project
                - -*
        - Action:
            - proton:UpdateServiceInstance
            - proton:GetServiceInstance
          Effect: Allow
          Resource: "*"
        - Action:
            - s3:GetObject*
            - s3:GetBucket*
            - s3:List*
          Effect: Allow
          Resource:
            - Fn::GetAtt:
                - PipelineArtifactsBucket
                - Arn
            - Fn::Join:
                - ""
                - - Fn::GetAtt:
                      - PipelineArtifactsBucket
                      - Arn
                  - /*
        - Action:
            - kms:Decrypt
            - kms:DescribeKey
          Effect: Allow
          Resource:
            Fn::GetAtt:
              - PipelineArtifactsBucketEncryptionKey
              - Arn
        - Action:
            - kms:Decrypt
            - kms:Encrypt
            - kms:ReEncrypt*
            - kms:GenerateDataKey*
          Effect: Allow
```

```
            Resource:
              Fn::GetAtt:
                - PipelineArtifactsBucketEncryptionKey
                - Arn
        Version: "2012-10-17"
      PolicyName: DeploymentRoleDefaultPolicy
      Roles:
        - Ref: DeploymentRole
  PipelineArtifactsBucketEncryptionKey:
    Type: AWS::KMS::Key
    Properties:
      KeyPolicy:
        Statement:
          - Action:
              - kms:Create*
              - kms:Describe*
              - kms:Enable*
              - kms:List*
              - kms:Put*
              - kms:Update*
              - kms:Revoke*
              - kms:Disable*
              - kms:Get*
              - kms:Delete*
              - kms:ScheduleKeyDeletion
              - kms:CancelKeyDeletion
              - kms:GenerateDataKey
              - kms:TagResource
              - kms:UntagResource
            Effect: Allow
            Principal:
              AWS:
                Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":iam::"
                    - Ref: AWS::AccountId
                    - :root
            Resource: "*"
          - Action:
              - kms:Decrypt
              - kms:DescribeKey
              - kms:Encrypt
```

```
            - kms:ReEncrypt*
            - kms:GenerateDataKey*
          Effect: Allow
          Principal:
            AWS:
              Fn::GetAtt:
                - PipelineRole
                - Arn
          Resource: "*"
        - Action:
            - kms:Decrypt
            - kms:DescribeKey
            - kms:Encrypt
            - kms:ReEncrypt*
            - kms:GenerateDataKey*
          Effect: Allow
          Principal:
            AWS:
              Fn::GetAtt:
                - PublishRole
                - Arn
          Resource: "*"
        - Action:
            - kms:Decrypt
            - kms:Encrypt
            - kms:ReEncrypt*
            - kms:GenerateDataKey*
          Effect: Allow
          Principal:
            AWS:
              Fn::GetAtt:
                - PublishRole
                - Arn
          Resource: "*"
        - Action:
            - kms:Decrypt
            - kms:DescribeKey
          Effect: Allow
          Principal:
            AWS:
              Fn::GetAtt:
                - DeploymentRole
                - Arn
          Resource: "*"
```

```
          - Action:
              - kms:Decrypt
              - kms:Encrypt
              - kms:ReEncrypt*
              - kms:GenerateDataKey*
            Effect: Allow
            Principal:
              AWS:
                Fn::GetAtt:
                  - DeploymentRole
                  - Arn
            Resource: "*"
        Version: "2012-10-17"
    UpdateReplacePolicy: Delete
    DeletionPolicy: Delete
  PipelineArtifactsBucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketEncryption:
        ServerSideEncryptionConfiguration:
          - ServerSideEncryptionByDefault:
              KMSMasterKeyID:
                Fn::GetAtt:
                  - PipelineArtifactsBucketEncryptionKey
                  - Arn
              SSEAlgorithm: aws:kms
      PublicAccessBlockConfiguration:
        BlockPublicAcls: true
        BlockPublicPolicy: true
        IgnorePublicAcls: true
        RestrictPublicBuckets: true
    UpdateReplacePolicy: Retain
    DeletionPolicy: Retain
  PipelineArtifactsBucketEncryptionKeyAlias:
    Type: AWS::KMS::Alias
    Properties:
      AliasName: 'alias/codepipeline-encryption-key-{{ service.name }}'     # resource
parameter
      TargetKeyId:
        Fn::GetAtt:
          - PipelineArtifactsBucketEncryptionKey
          - Arn
    UpdateReplacePolicy: Delete
    DeletionPolicy: Delete
```

```
  PipelineRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              Service: codepipeline.amazonaws.com
        Version: "2012-10-17"
  PipelineRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - s3:GetObject*
              - s3:GetBucket*
              - s3:List*
              - s3:DeleteObject*
              - s3:PutObject*
              - s3:Abort*
            Effect: Allow
            Resource:
              - Fn::GetAtt:
                  - PipelineArtifactsBucket
                  - Arn
              - Fn::Join:
                  - ""
                  - - Fn::GetAtt:
                        - PipelineArtifactsBucket
                        - Arn
                    - /*
          - Action:
              - kms:Decrypt
              - kms:DescribeKey
              - kms:Encrypt
              - kms:ReEncrypt*
              - kms:GenerateDataKey*
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineArtifactsBucketEncryptionKey
                - Arn
```

```
          - Action: codestar-connections:*
            Effect: Allow
            Resource: "*"
          - Action: sts:AssumeRole
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineBuildCodePipelineActionRole
                - Arn
          - Action: sts:AssumeRole
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - PipelineDeployCodePipelineActionRole
                - Arn
        Version: "2012-10-17"
      PolicyName: PipelineRoleDefaultPolicy
      Roles:
        - Ref: PipelineRole
  Pipeline:
    Type: AWS::CodePipeline::Pipeline
    Properties:
      RoleArn:
        Fn::GetAtt:
          - PipelineRole
          - Arn
      Stages:
        - Actions:
            - ActionTypeId:
                Category: Source
                Owner: AWS
                Provider: CodeStarSourceConnection
                Version: "1"
              Configuration:
                ConnectionArn: '{{ service.repository_connection_arn }}'   # resource
parameter
                FullRepositoryId: '{{ service.repository_id }}'            # resource
parameter
                BranchName: '{{ service.branch_name }}'                    # resource
parameter
              Name: Checkout
              OutputArtifacts:
                - Name: Artifact_Source_Checkout
              RunOrder: 1
```

```
          Name: Source
        - Actions:
            - ActionTypeId:
                Category: Build
                Owner: AWS
                Provider: CodeBuild
                Version: "1"
              Configuration:
                ProjectName:
                  Ref: BuildProject
              InputArtifacts:
                - Name: Artifact_Source_Checkout
              Name: Build
              OutputArtifacts:
                - Name: BuildOutput
              RoleArn:
                Fn::GetAtt:
                  - PipelineBuildCodePipelineActionRole
                  - Arn
              RunOrder: 1
          Name: Build {%- for service_instance in service_instances %}
        - Actions:
            - ActionTypeId:
                Category: Build
                Owner: AWS
                Provider: CodeBuild
                Version: "1"
              Configuration:
                ProjectName:
                  Ref: Deploy{{loop.index}}Project
              InputArtifacts:
                - Name: BuildOutput
              Name: Deploy
              RoleArn:
                Fn::GetAtt:
                  - PipelineDeployCodePipelineActionRole
                  - Arn
              RunOrder: 1
          Name: 'Deploy{{service_instance.name}}'         # resource parameter
{%- endfor %}
      ArtifactStore:
        EncryptionKey:
          Id:
            Fn::GetAtt:
```

```
              - PipelineArtifactsBucketEncryptionKey
              - Arn
          Type: KMS
        Location:
          Ref: PipelineArtifactsBucket
        Type: S3
    DependsOn:
      - PipelineRoleDefaultPolicy
      - PipelineRole
  PipelineBuildCodePipelineActionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              AWS:
                Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":iam::"
                    - Ref: AWS::AccountId
                    - :root
        Version: "2012-10-17"
  PipelineBuildCodePipelineActionRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - codebuild:BatchGetBuilds
              - codebuild:StartBuild
              - codebuild:StopBuild
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - BuildProject
                - Arn
        Version: "2012-10-17"
      PolicyName: PipelineBuildCodePipelineActionRoleDefaultPolicy
      Roles:
        - Ref: PipelineBuildCodePipelineActionRole
```

```
  PipelineDeployCodePipelineActionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              AWS:
                Fn::Join:
                  - ""
                  - - "arn:"
                    - Ref: AWS::Partition
                    - ":iam::"
                    - Ref: AWS::AccountId
                    - :root
        Version: "2012-10-17"
  PipelineDeployCodePipelineActionRoleDefaultPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action:
              - codebuild:BatchGetBuilds
              - codebuild:StartBuild
              - codebuild:StopBuild
            Effect: Allow
            Resource:
              Fn::Join:
                - ""
                - - "arn:"
                  - Ref: AWS::Partition
                  - ":codebuild:"
                  - Ref: AWS::Region
                  - ":"
                  - Ref: AWS::AccountId
                  - ":project/Deploy*"
        Version: "2012-10-17"
      PolicyName: PipelineDeployCodePipelineActionRoleDefaultPolicy
      Roles:
        - Ref: PipelineDeployCodePipelineActionRole
Outputs:
  PipelineEndpoint:
    Description: The URL to access the pipeline
```

```
    Value: !Sub "https://${AWS::Region}.console.aws.amazon.com/codesuite/codepipeline/
pipelines/${Pipeline}/view?region=${AWS::Region}"
```

CodeBuild bundel templat penyediaan

Dengan CodeBuild penyediaan, alih-alih menggunakan templat IAc untuk merender file IAc dan menjalankannya menggunakan mesin penyediaan IAC, cukup jalankan perintah shell Anda. AWS Proton Untuk melakukan itu, AWS Proton buat AWS CodeBuild proyek untuk lingkungan, di akun lingkungan, dan mulai pekerjaan untuk menjalankan perintah Anda untuk setiap pembuatan atau pembaruan AWS Proton sumber daya. Saat Anda membuat bundel templat, Anda menyediakan manifes yang menentukan perintah penyediaan dan penonjolan infrastruktur, serta program, skrip, dan file lain yang mungkin diperlukan oleh perintah ini. Perintah Anda dapat membaca input yang AWS Proton menyediakan, dan bertanggung jawab untuk penyediaan atau deprovisioning infrastruktur dan menghasilkan nilai output.

Manifes juga menentukan bagaimana AWS Proton seharusnya merender file input yang kode Anda dapat input dan mendapatkan nilai masukan dari. Itu dapat dirender ke JSON atau HCL. Untuk informasi selengkapnya tentang parameter input, lihatthe section called “CodeBuild parameter penyediaan”. Untuk informasi selengkapnya tentang file manifes, lihatthe section called “Manifestasikan dan bungkus”.

> **Note**
>
> Anda dapat menggunakan CodeBuild penyediaan dengan lingkungan dan layanan. Saat ini Anda tidak dapat menyediakan komponen dengan cara ini.

Contoh: menggunakan CodeBuild penyediaan AWS CDK with

Sebagai contoh untuk menggunakan CodeBuild provisioning, Anda dapat menyertakan kode yang menggunakan AWS sumber daya AWS Cloud Development Kit (AWS CDK) to provisioning (deploy) dan deprovision (destroy), serta manifes yang menginstal CDK dan menjalankan kode CDK Anda.

Bagian berikut mencantumkan file contoh yang dapat Anda sertakan dalam bundel templat CodeBuild penyediaan yang menyediakan lingkungan menggunakan. AWS CDK

Manifes

File manifes berikut CodeBuild menentukan penyediaan, dan menyertakan perintah yang diperlukan untuk menginstal dan menggunakan, pemrosesan file keluaran AWS CDK, dan pelaporan output kembali ke. AWS Proton

Example infrastruktur/manifest.yaml

```
infrastructure:
  templates:
    - rendering_engine: codebuild
      settings:
        image: aws/codebuild/amazonlinux2-x86_64-standard:4.0
        runtimes:
          nodejs: 16
        provision:
          - npm install
          - npm run build
          - npm run cdk bootstrap
          - npm run cdk deploy -- --require-approval never --outputs-file proton-
outputs.json
          - jq 'to_entries | map_values(.value) | add | to_entries | map({key:.key,
 valueString:.value})' < proton-outputs.json > outputs.json
          - aws proton notify-resource-deployment-status-change --resource-arn
 $RESOURCE_ARN --status IN_PROGRESS --outputs file://./outputs.json
        deprovision:
          - npm install
          - npm run build
          - npm run cdk destroy
        project_properties:
          VpcConfig:
            VpcId: "{{ environment.inputs.codebuild_vpc_id }}"
            Subnets: "{{ environment.inputs.codebuild_subnets }}"
            SecurityGroupIds: "{{ environment.inputs.codebuild_security_groups }}"
```

Skema

File skema berikut mendefinisikan parameter untuk lingkungan. AWS CDK Kode Anda dapat merujuk ke nilai parameter ini selama penerapan.

Example schema/schema.yaml

```
schema:
```

```
  format:
    openapi: "3.0.0"
  environment_input_type: "MyEnvironmentInputType"
  types:
    MyEnvironmentInputType:
      type: object
      description: "Input properties for my environment"
      properties:
        my_sample_input:
          type: string
          description: "This is a sample input"
          default: "hello world"
        my_other_sample_input:
          type: string
          description: "Another sample input"
      required:
        - my_other_sample_input
```

AWS CDK berkas

File-file berikut adalah contoh untuk proyek CDK Node.js.

Example infrastruktur/package.json

```
{
  "name": "ProtonEnvironment",
  "version": "0.1.0",
  "bin": {
    "ProtonEnvironmente": "bin/ProtonEnvironment.js"
  },
  "scripts": {
    "build": "tsc",
    "watch": "tsc -w",
    "test": "jest",
    "cdk": "cdk"
  },
  "devDependencies": {
    "@types/jest": "^28.1.7",
    "@types/node": "18.7.6",
    "jest": "^28.1.3",
    "ts-jest": "^28.0.8",
    "aws-cdk": "2.37.1",
    "ts-node": "^10.9.1",
```

```
    "typescript": "~4.7.4"
  },
  "dependencies": {
    "aws-cdk-lib": "2.37.1",
    "constructs": "^10.1.77",
    "source-map-support": "^0.5.21"
  }
}
```

Example infrastruktur/tsconfig.json

```
{
  "compilerOptions": {
    "target": "ES2018",
    "module": "commonjs",
    "lib": [
      "es2018"
    ],
    "declaration": true,
    "strict": true,
    "noImplicitAny": true,
    "strictNullChecks": true,
    "noImplicitThis": true,
    "alwaysStrict": true,
    "noUnusedLocals": false,
    "noUnusedParameters": false,
    "noImplicitReturns": true,
    "noFallthroughCasesInSwitch": false,
    "inlineSourceMap": true,
    "inlineSources": true,
    "experimentalDecorators": true,
    "strictPropertyInitialization": false,
    "resolveJsonModule": true,
    "esModuleInterop": true,
    "typeRoots": [
      "./node_modules/@types"
    ]
  },
  "exclude": [
    "node_modules",
    "cdk.out"
  ]
}
```

Example infrastruktur/cdk.json

```
{
  "app": "npx ts-node --prefer-ts-exts bin/ProtonEnvironment.ts",
  "outputsFile": "proton-outputs.json",
  "watch": {
    "include": [
      "**"
    ],
    "exclude": [
      "README.md",
      "cdk*.json",
      "**/*.d.ts",
      "**/*.js",
      "tsconfig.json",
      "package*.json",
      "yarn.lock",
      "node_modules",
      "test"
    ]
  },
  "context": {
    "@aws-cdk/aws-apigateway:usagePlanKeyOrderInsensitiveId": true,
    "@aws-cdk/core:stackRelativeExports": true,
    "@aws-cdk/aws-rds:lowercaseDbIdentifier": true,
    "@aws-cdk/aws-lambda:recognizeVersionProps": true,
    "@aws-cdk/aws-cloudfront:defaultSecurityPolicyTLSv1.2_2021": true,
    "@aws-cdk-containers/ecs-service-extensions:enableDefaultLogDriver": true,
    "@aws-cdk/aws-ec2:uniqueImdsv2TemplateName": true,
    "@aws-cdk/core:target-partitions": [
      "aws",
      "aws-cn"
    ]
  }
}
```

Example infrastructure/bin/ProtonEnvironment.ts

```
#!/usr/bin/env node
import 'source-map-support/register';
import * as cdk from 'aws-cdk-lib';
import { ProtonEnvironmentStack } from '../lib/ProtonEnvironmentStack';
```

```
const app = new cdk.App();
new ProtonEnvironmentStack(app, 'ProtonEnvironmentStack', {});
```

Example infrastructure/lib/ProtonEnvironmentStack.ts

```
import { Stack, StackProps } from 'aws-cdk-lib';
import { Construct } from 'constructs';
import * as cdk from 'aws-cdk-lib';
import * as ssm from 'aws-cdk-lib/aws-ssm';
import input from '../proton-inputs.json';

export class ProtonEnvironmentStack extends Stack {
  constructor(scope: Construct, id: string, props?: StackProps) {
    super(scope, id, { ...props, stackName: process.env.STACK_NAME });

    const ssmParam = new ssm.StringParameter(this, "ssmParam", {
      stringValue: input.environment.inputs.my_sample_input,
      parameterName: `${process.env.STACK_NAME}-Param`,
      tier: ssm.ParameterTier.STANDARD
    })

    new cdk.CfnOutput(this, 'ssmParamOutput', {
      value: ssmParam.parameterName,
      description: 'The name of the ssm parameter',
      exportName: `${process.env.STACK_NAME}-Param`
    });
  }
}
```

File masukan yang dirender

Saat Anda membuat lingkungan menggunakan templat penyediaan CodeBuild berbasis, AWS Proton merender file input dengan nilai parameter masukan yang Anda berikan. Kode Anda dapat merujuk ke nilai-nilai ini. File berikut adalah contoh untuk file input yang diberikan.

Example infrastruktur/proton-inputs.json

```
{
  "environment": {
    "name": "myenv",
    "inputs": {
      "my_sample_input": "10.0.0.0/16",
```

```
      "my_other_sample_input": "11.0.0.0/16"
    }
  }
}
```

File Terraform IAc

Pelajari cara menggunakan infrastruktur Terraform sebagai file kode (IAc) dengan. AWS ProtonTerraform adalah mesin iAC open-source yang banyak digunakan yang dikembangkan oleh. HashiCorp Modul Terraform dikembangkan dalam bahasa HashiCorp HCL, dan mendukung beberapa penyedia infrastruktur backend, termasuk Amazon Web Services.

AWS Proton mendukung penyediaan yang dikelola sendiri untuk Terraform IAc.

Untuk contoh lengkap repositori penyediaan yang merespons permintaan tarik dan mengimplementasikan penyediaan infrastruktur, lihat Templat otomatisasi Tindakan Terraform untuk on. OpenSource GitHub AWS Proton GitHub

Cara kerja penyediaan yang dikelola sendiri dengan file bundel template Terraform IAc:

1. Saat Anda membuat lingkungan dari bundel template Terraform, AWS Proton kompilasi `.tf` file Anda dengan parameter konsol atau input. `spec file`
2. Itu membuat permintaan tarik untuk menggabungkan file IAC yang dikompilasi ke repositori yang telah Anda daftarkan. AWS Proton
3. Jika permintaan disetujui, AWS Proton tunggu status penyediaan yang Anda berikan.
4. Jika permintaan ditolak, pembuatan lingkungan dibatalkan.
5. Jika waktu permintaan tarik habis, pembuatan lingkungan tidak selesai.

AWS Proton dengan pertimbangan Terraform IAc:

- AWS Proton tidak mengelola penyediaan Terraform Anda.
- Anda harus mendaftarkan repositori penyediaan dengan. AWS Proton AWS Proton membuat permintaan tarik pada repositori ini.
- Anda harus membuat CodeStar koneksi untuk terhubung AWS Proton dengan repositori penyediaan Anda.
- Untuk menyediakan dari file IAc yang AWS Proton dikompilasi, Anda harus menanggapi permintaan AWS Proton tarik. AWS Proton membuat permintaan tarik setelah lingkungan dan

layanan membuat dan memperbarui tindakan. Untuk informasi selengkapnya, lihat AWS Proton lingkungan dan AWS Proton layanan.

- Untuk menyediakan pipeline dari file IAc yang AWS Proton dikompilasi, Anda harus membuat repositori CI/CD pipeline.
- Otomatisasi penyediaan berbasis permintaan tarik Anda harus menyertakan langkah-langkah untuk memberi tahu perubahan status AWS Proton sumber daya yang disediakan AWS Proton . Anda dapat menggunakan AWS Proton NotifyResourceDeploymentStatusChange API.
- Anda tidak dapat menerapkan layanan, saluran pipa, dan komponen yang dibuat dari file CloudFormation IAc ke lingkungan yang dibuat dari file Terraform IAc.
- Anda tidak dapat menerapkan layanan, saluran pipa, dan komponen yang dibuat dari file Terraform IAc ke lingkungan yang dibuat dari file IAc. CloudFormation

Saat menyiapkan file Terraform IAc Anda AWS Proton, Anda melampirkan ruang nama ke variabel input Anda, seperti yang ditunjukkan pada contoh berikut. Untuk informasi lebih lanjut, lihat Parameter.

Contoh 1: AWS Proton lingkungan file Terraform IAc

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 3.0"
    }
  }
  // This tells terraform to store the state file in s3 at the location
  // s3://terraform-state-bucket/tf-os-sample/terraform.tfstate
  backend "s3" {
    bucket = "terraform-state-bucket"
    key    = "tf-os-sample/terraform.tfstate"
    region = "us-east-1"
  }
}

// Configure the AWS Provider
provider "aws" {
  region = "us-east-1"
  default_tags {
    tags = var.proton_tags
  }
```

```
}

resource "aws_ssm_parameter" "my_ssm_parameter" {
  name  = "my_ssm_parameter"
  type  = "String"
  // Use the Proton environment.inputs. namespace
  value = var.environment.inputs.ssm_parameter_value
}
```

Infrastruktur yang dikompilasi sebagai kode

Saat Anda membuat lingkungan atau layanan, AWS Proton kompilasi infrastruktur Anda sebagai file kode dengan konsol atau `spec file` input. Ini membuat `proton.`*`resource-type`*`.variables.tf` dan `proton.auto.tfvars.json` file untuk input Anda yang dapat digunakan oleh Terraform, seperti yang ditunjukkan dalam contoh berikut. File-file ini terletak di repositori tertentu di folder yang cocok dengan nama instance lingkungan atau layanan.

Contoh ini menunjukkan bagaimana AWS Proton menyertakan tag dalam definisi variabel dan nilai variabel, dan bagaimana Anda dapat menyebarkan AWS Proton tag ini ke sumber daya yang disediakan. Untuk informasi selengkapnya, lihat the section called “Perbanyakan tag ke sumber daya yang disediakan”.

Contoh 2: file IAc yang dikompilasi untuk lingkungan bernama “dev”.

dev/environment.tf:

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 3.0"
    }
  }
  // This tells terraform to store the state file in s3 at the location
  // s3://terraform-state-bucket/tf-os-sample/terraform.tfstate
  backend "s3" {
    bucket = "terraform-state-bucket"
    key    = "tf-os-sample/terraform.tfstate"
    region = "us-east-1"
  }
}
```

```
// Configure the AWS Provider
provider "aws" {
  region = "us-east-1"
  default_tags {
    tags = var.proton_tags
  }
}

resource "aws_ssm_parameter" "my_ssm_parameter" {
  name  = "my_ssm_parameter"
  type  = "String"
  // Use the Proton environment.inputs. namespace
  value = var.environment.inputs.ssm_parameter_value
}
```

dev/proton.environment.variables.tf:

```
variable "environment" {
  type = object({
    inputs = map(string)
    name = string
  })
}

variable "proton_tags" {
  type = map(string)
  default = null
}
```

dev/proton.auto.tfvars.json:

```
{
  "environment": {
    "name": "dev",
    "inputs": {
      "ssm_parameter_value": "MyNewParamValue"
    }
  }

  "proton_tags" : {
    "proton:account" : "123456789012",
    "proton:template" : "arn:aws:proton:us-east-1:123456789012:environment-template/
fargate-env",
```

```
    "proton:environment" : "arn:aws:proton:us-east-1:123456789012:environment/dev"
  }
}
```

Jalur repositori

AWS Proton menggunakan input konsol atau spesifikasi dari tindakan pembuatan lingkungan atau layanan untuk menemukan repositori dan jalur tempat untuk menemukan file IAc yang dikompilasi. Nilai masukan diteruskan ke parameter masukan namespaced.

AWS Proton mendukung dua tata letak jalur repositori. Dalam contoh berikut, jalur diberi nama oleh parameter sumber daya namespaced dari dua lingkungan. Setiap lingkungan memiliki contoh layanan dari dua layanan, dan contoh layanan dari salah satu layanan memiliki komponen yang didefinisikan secara langsung.

Jenis sumber daya	Parameter nama	=	Nama sumber daya
Lingkungan	`environment.name`		`"env-prod"`
Lingkungan	`environment.name`		`"env-staged"`
Layanan	`service.name`		`"service-one"`
Contoh layanan	`service_instance.name`	=	`"instance-one-prod"`
Contoh layanan	`service_instance.name`		`"instance-one-staged"`
Layanan	`service.name`		`"service-two"`

Jenis sumber daya	Parameter nama	=	Nama sumber daya
Contoh layanan	`service_instance.name`		`"instance-two-prod"`
Komponen	`service_instance.components.default.name`		`"component-prod"`
Contoh layanan	`service_instance.name`		`"instance-two-staged"`
Komponen	`service_instance.components.default.name`		`"component-staged"`

Layout 1

Jika AWS Proton menemukan repositori yang ditentukan dengan `environments` folder, itu membuat folder yang menyertakan file IAC yang dikompilasi dan diberi nama dengan file. `environment.name`

Jika AWS Proton menemukan repositori yang ditentukan dengan `environments` folder yang berisi nama folder yang cocok dengan nama lingkungan yang kompatibel dengan instance layanan, itu membuat folder yang menyertakan file iAc instance yang dikompilasi dan diberi nama dengan file. `service_instance.name`

```
/repo
    /environments
        /env-prod                               # environment folder
            main.tf
            proton.environment.variables.tf
            proton.auto.tfvars.json

            /service-one-instance-one-prod      # instance folder
                main.tf
```

```
            proton.service_instance.variables.tf
            proton.auto.tfvars.json

        /service-two-instance-two-prod    # instance folder
            main.tf
            proton.service_instance.variables.tf
            proton.auto.tfvars.json

        /component-prod                   # component folder
            main.tf
            proton.component.variables.tf
            proton.auto.tfvars.json

    /env-staged                           # environment folder
        main.tf
        proton.variables.tf
        proton.auto.tfvars.json

        /service-one-instance-one-staged  # instance folder
            main.tf
            proton.service_instance.variables.tf
            proton.auto.tfvars.json

        /service-two-instance-two-staged  # instance folder
            main.tf
            proton.service_instance.variables.tf
            proton.auto.tfvars.json

        /component-staged                 # component folder
            main.tf
            proton.component.variables.tf
            proton.auto.tfvars.json
```

Layout 2

Jika AWS Proton menemukan repositori yang ditentukan tanpa `environments` folder, itu membuat `environment.name` folder di mana ia menemukan file IAc lingkungan yang dikompilasi.

Jika AWS Proton menemukan repositori yang ditentukan dengan nama folder yang cocok dengan nama lingkungan yang kompatibel dengan instance layanan, itu akan membuat `service_instance.name` folder tempat ia menempatkan file iAc instance yang dikompilasi.

```
/repo
    /env-prod                              # environment folder
        main.tf
        proton.environment.variables.tf
        proton.auto.tfvars.json

        /service-one-instance-one-prod     # instance folder
            main.tf
            proton.service_instance.variables.tf
            proton.auto.tfvars.json

        /service-two-instance-two-prod     # instance folder
            main.tf
            proton.service_instance.variables.tf
            proton.auto.tfvars.json

        /component-prod                    # component folder
            main.tf
            proton.component.variables.tf
            proton.auto.tfvars.json

    /env-staged                            # environment folder
        main.tf
        proton.variables.tf
        proton.auto.tfvars.json

        /service-one-instance-one-staged   # instance folder
            main.tf
            proton.service_instance.variables.tf
            proton.auto.tfvars.json

        /service-two-instance-two-staged   # instance folder
            main.tf
            proton.service_instance.variables.tf
            proton.auto.tfvars.json

        /component-staged                  # component folder
            main.tf
            proton.component.variables.tf
            proton.auto.tfvars.json
```

Berkas skema

Sebagai administrator, saat Anda menggunakan bagian Open API Data Models (schemas) untuk menentukan skema parameter file YAMB untuk paket template Anda, AWS Proton dapat memvalidasi input nilai parameter terhadap persyaratan yang Anda tetapkan dalam skema Anda.

Untuk informasi selengkapnya tentang format dan kata kunci yang tersedia, lihat bagian objek Skema di OpenAPI.

Persyaratan skema untuk bundel templat lingkungan

Skema Anda harus mengikuti bagian Model Data (skema) OpenAPI dalam format YAMAL. Itu juga harus menjadi bagian dari bundel template lingkungan Anda.

Untuk skema lingkungan Anda, Anda harus menyertakan header yang diformat untuk menetapkan bahwa Anda menggunakan bagian Model Data (skema) dari Open API. Dalam contoh skema lingkungan berikut, header ini muncul di tiga baris pertama.

An `environment_input_type` harus disertakan dan didefinisikan dengan nama yang Anda berikan. Dalam contoh berikut, ini didefinisikan pada baris 5. Dengan mendefinisikan parameter ini, Anda mengaitkannya dengan sumber daya AWS Proton lingkungan.

Untuk mengikuti model skema Open API, Anda harus menyertakan`types`. Dalam contoh berikut, ini adalah baris 6.

Berikut`types`, Anda harus menentukan `environment_input_type` tipe. Anda menentukan parameter input untuk lingkungan Anda sebagai properti dari`environment_input_type`. Anda harus menyertakan setidaknya satu properti dengan nama yang cocok dengan setidaknya satu parameter yang tercantum dalam infrastruktur lingkungan sebagai file kode (IAc) yang terkait dengan skema.

Saat Anda membuat lingkungan dan memberikan nilai parameter yang disesuaikan, AWS Proton gunakan file skema untuk mencocokkan, memvalidasi, dan menyuntikkannya ke dalam parameter kurawal kurawal dalam file IAc terkait. CloudFormation Untuk setiap properti (parameter), berikan `name` dan`type`. Secara opsional, juga menyediakan`description`,`default`, dan`pattern`.

Parameter yang ditentukan untuk contoh skema template lingkungan standar berikut meliputi`vpc_cidr`,`subnet_one_cidr`, dan `subnet_two_cidr` dengan `default` kata kunci dan nilai default. Saat Anda membuat lingkungan dengan skema bundel template lingkungan ini, Anda dapat menerima nilai default atau memberikan nilai Anda sendiri. Jika parameter tidak memiliki nilai

default dan terdaftar sebagai `required` properti (parameter), Anda harus memberikan nilai untuk itu ketika Anda membuat lingkungan.

Contoh kedua skema template lingkungan standar mencantumkan `required` parameter`my_other_sample_input`.

Anda dapat membuat skema untuk dua jenis template lingkungan. Untuk informasi selengkapnya, lihat Daftarkan dan terbitkan templat.

- Template lingkungan standar

  Dalam contoh berikut, jenis input lingkungan didefinisikan dengan deskripsi dan properti input. Contoh skema ini dapat digunakan dengan file AWS Proton CloudFormation IAc yang ditunjukkan pada Contoh 3.

  Contoh skema untuk template lingkungan standar:

```
schema:                           # required
  format:                         # required
    openapi: "3.0.0"              # required
  # required              defined by administrator
  environment_input_type: "PublicEnvironmentInput"
  types:                          # required
    # defined by administrator
    PublicEnvironmentInput:
      type: object
      description: "Input properties for my environment"
      properties:
        vpc_cidr:                  # parameter
          type: string
          description: "This CIDR range for your VPC"
          default: 10.0.0.0/16
          pattern: ([0-9]{1,3}\.){3}[0-9]{1,3}($|/(16|24))
        subnet_one_cidr:           # parameter
          type: string
          description: "The CIDR range for subnet one"
          default: 10.0.0.0/24
          pattern: ([0-9]{1,3}\.){3}[0-9]{1,3}($|/(16|24))
        subnet_two_cidr:           # parameter
          type: string
          description: "The CIDR range for subnet one"
          default: 10.0.1.0/24
```

```
          pattern: ([0-9]{1,3}\.){3}[0-9]{1,3}($|/(16|24))
```

Contoh skema untuk template lingkungan standar yang menyertakan `required` parameter:

```
schema:                           # required
  format:                         # required
    openapi: "3.0.0"              # required
  # required              defined by administrator
  environment_input_type: "MyEnvironmentInputType"
  types:                          # required
    # defined by administrator
    MyEnvironmentInputType:
      type: object
      description: "Input properties for my environment"
      properties:
        my_sample_input:          # parameter
          type: string
          description: "This is a sample input"
          default: "hello world"
        my_other_sample_input:    # parameter
          type: string
          description: "Another sample input"
        another_optional_input:   # parameter
          type: string
          description: "Another optional input"
          default: "!"
      required:
        - my_other_sample_input
```

- Templat lingkungan yang dikelola pelanggan

  Dalam contoh berikut, skema hanya menyertakan daftar output yang mereplikasi output dari IAc yang Anda gunakan untuk menyediakan infrastruktur yang dikelola pelanggan Anda. Anda perlu mendefinisikan jenis nilai output sebagai string saja (bukan daftar, array atau tipe lainnya). Misalnya, cuplikan kode berikutnya menunjukkan bagian output dari template eksternal. CloudFormation Ini dari template yang ditunjukkan pada Contoh 1. Ini dapat digunakan untuk membuat infrastruktur yang dikelola pelanggan eksternal untuk layanan AWS Proton Fargate yang dibuat dari Contoh 4.

> **Important**
>
> Sebagai administrator, Anda harus memastikan bahwa infrastruktur yang disediakan dan dikelola serta semua parameter keluaran kompatibel dengan templat lingkungan terkelola pelanggan terkait. AWS Proton tidak dapat memperhitungkan perubahan atas nama Anda karena perubahan ini tidak terlihat AWS Proton. Ketidakkonsistenan mengakibatkan kegagalan.

Contoh output file CloudFormation IAC untuk template lingkungan yang dikelola pelanggan:

```
// Cloudformation Template Outputs
[...]
Outputs:
  ClusterName:
    Description: The name of the ECS cluster
    Value: !Ref 'ECSCluster'
  ECSTaskExecutionRole:
    Description: The ARN of the ECS role
    Value: !GetAtt 'ECSTaskExecutionRole.Arn'
  VpcId:
    Description: The ID of the VPC that this stack is deployed in
    Value: !Ref 'VPC'
[...]
```

Skema untuk bundel template lingkungan terkelola AWS Proton pelanggan yang sesuai ditunjukkan dalam contoh berikut. Setiap nilai output didefinisikan sebagai string.

Contoh skema untuk template lingkungan yang dikelola pelanggan:

```
schema:                           # required
  format:                         # required
    openapi: "3.0.0"              # required
  # required              defined by administrator
  environment_input_type: "EnvironmentOutput"
  types:                          # required
    # defined by administrator
    EnvironmentOutput:
      type: object
      description: "Outputs of the environment"
```

```
      properties:
        ClusterName:               # parameter
          type: string
          description: "The name of the ECS cluster"
        ECSTaskExecutionRole:      # parameter
          type: string
          description: "The ARN of the ECS role"
        VpcId:                     # parameter
          type: string
          description: "The ID of the VPC that this stack is deployed in"
[...]
```

Persyaratan skema untuk bundel templat layanan

Skema Anda harus mengikuti bagian Model Data (skema) OpenAPI dalam format YAMAL seperti yang ditunjukkan pada contoh berikut. Anda harus menyediakan file skema dalam paket template layanan Anda.

Dalam contoh skema layanan berikut, Anda harus menyertakan header yang diformat. Dalam contoh berikut, ini ada di tiga baris pertama. Ini untuk menetapkan bahwa Anda menggunakan bagian Model Data (skema) dari Open API.

A `service_input_type` harus disertakan dan didefinisikan dengan nama yang Anda berikan. Dalam contoh berikut, ini ada di baris 5. Ini mengaitkan parameter dengan sumber daya AWS Proton layanan.

Pipeline AWS Proton layanan disertakan secara default saat Anda menggunakan konsol atau CLI untuk membuat layanan. Ketika Anda menyertakan saluran layanan untuk layanan Anda, Anda harus menyertakan `pipeline_input_type` dengan nama yang Anda berikan. Dalam contoh berikut, ini ada di baris 7. Jangan sertakan parameter ini jika Anda tidak menyertakan pipeline AWS Proton layanan. Untuk informasi selengkapnya, lihat Daftarkan dan terbitkan templat.

Untuk mengikuti model skema Open API, Anda harus menyertakan `types` Dalam contoh berikut, ini ada di baris 9.

Berikut`types`, Anda harus menentukan `service_input_type` tipe. Anda menentukan parameter input untuk layanan Anda sebagai properti dari`service_input_type`. Anda harus menyertakan setidaknya satu properti dengan nama yang cocok dengan setidaknya satu parameter yang tercantum dalam file Service Infrastructure as code (IAc) yang terkait dengan skema.

Untuk menentukan pipeline layanan, di bawah `service_input_type` definisi Anda, Anda harus menentukan`pipeline_input_type`. Seperti di atas, Anda harus menyertakan setidaknya satu properti dengan nama yang cocok dengan setidaknya satu parameter yang tercantum dalam file iAc pipeline yang terkait dengan skema. Jangan sertakan definisi ini jika Anda tidak menyertakan pipeline AWS Proton layanan.

Saat Anda, sebagai administrator atau pengembang, membuat layanan dan memberikan nilai parameter yang disesuaikan, AWS Proton menggunakan file skema untuk mencocokkan, memvalidasi, dan menyuntikkannya ke dalam parameter kurung kurawal file CloudFormation IAC terkait. Untuk setiap properti (parameter), berikan a `name` dan `atype`. Secara opsional, juga menyediakan`description`,`default`, dan`pattern`.

Parameter yang ditentukan untuk skema contoh meliputi`port`,`desired_count`, `task_size` dan `image` dengan `default` kata kunci dan nilai default. Saat Anda membuat layanan dengan skema bundel template layanan ini, Anda dapat menerima nilai default atau memberikan nilai Anda sendiri. Parameter `unique_name` ini juga disertakan dalam contoh dan tidak memiliki nilai default. Ini terdaftar sebagai `required` properti (parameter). Anda, sebagai administrator atau pengembang, harus memberikan nilai untuk `required` parameter saat Anda membuat layanan.

Jika Anda ingin membuat template layanan dengan pipeline layanan, sertakan `pipeline_input_type` dalam skema Anda.

Contoh file skema layanan untuk layanan yang menyertakan pipeline AWS Proton layanan.

Contoh skema ini dapat digunakan dengan file AWS Proton IAc yang ditunjukkan pada Contoh 4 dan Contoh 5. Pipa layanan disertakan.

```
schema:                           # required
  format:                         # required
    openapi: "3.0.0"              # required
  # required           defined by administrator
  service_input_type: "LoadBalancedServiceInput"
  # only include if including AWS Proton service pipeline, defined by administrator
  pipeline_input_type: "PipelineInputs"

  types:                          # required
    # defined by administrator
    LoadBalancedServiceInput:
      type: object
      description: "Input properties for a loadbalanced Fargate service"
      properties:
```

```
        port:                       # parameter
          type: number
          description: "The port to route traffic to"
          default: 80
          minimum: 0
          maximum: 65535
        desired_count:              # parameter
          type: number
          description: "The default number of Fargate tasks you want running"
          default: 1
          minimum: 1
        task_size:                  # parameter
          type: string
          description: "The size of the task you want to run"
          enum: ["x-small", "small", "medium", "large", "x-large"]
          default: "x-small"
        image:                      # parameter
          type: string
          description: "The name/url of the container image"
          default: "public.ecr.aws/z9d2n7e1/nginx:1.19.5"
          minLength: 1
          maxLength: 200
        unique_name:                # parameter
          type: string
          description: "The unique name of your service identifier. This will be used
to name your log group, task definition and ECS service"
          minLength: 1
          maxLength: 100
      required:
        - unique_name
    # defined by administrator
    PipelineInputs:
      type: object
      description: "Pipeline input properties"
      properties:
        dockerfile:                 # parameter
          type: string
          description: "The location of the Dockerfile to build"
          default: "Dockerfile"
          minLength: 1
          maxLength: 100
        unit_test_command:          # parameter
          type: string
          description: "The command to run to unit test the application code"
```

```
          default: "echo 'add your unit test command here'"
          minLength: 1
          maxLength: 200
```

Jika Anda ingin membuat template layanan tanpa pipeline layanan, jangan sertakan `pipeline_input_type` dalam skema Anda, seperti yang ditunjukkan pada contoh berikut.

Contoh file skema layanan untuk layanan yang tidak menyertakan pipeline AWS Proton layanan

```
schema:                           # required
  format:                         # required
    openapi: "3.0.0"              # required
  # required            defined by administrator
  service_input_type: "MyServiceInstanceInputType"

  types:                          # required
    # defined by administrator
    MyServiceInstanceInputType:
      type: object
      description: "Service instance input properties"
      required:
        - my_sample_service_instance_required_input
      properties:
        my_sample_service_instance_optional_input:   # parameter
          type: string
          description: "This is a sample input"
          default: "hello world"
        my_sample_service_instance_required_input:   # parameter
          type: string
          description: "Another sample input"
```

Bungkus file template untuk AWS Proton

Setelah menyiapkan infrastruktur lingkungan dan layanan Anda sebagai file kode (IAc) dan file skema masing-masing, Anda harus mengaturnya dalam direktori. Anda juga harus membuat file YAML manifes. File manifes mencantumkan file IAc dalam direktori, mesin rendering, dan bahasa template yang digunakan untuk mengembangkan IAc dalam template ini.

> **Note**
>
> File manifes juga dapat digunakan secara independen dari bundel template, sebagai input langsung ke komponen yang didefinisikan secara langsung. Dalam hal ini, selalu menentukan satu file template IAC, untuk keduanya CloudFormation dan Terraform. Untuk informasi selengkapnya tentang komponen, lihatKomponen-komponen.

File manifes harus mematuhi format dan konten yang ditunjukkan dalam contoh berikut.

CloudFormation format file manifes:

Dengan CloudFormation, Anda mencantumkan satu file.

```
infrastructure:
  templates:
    - file: "cloudformation.yaml"
      rendering_engine: jinja
      template_language: cloudformation
```

Format file manifes Terraform:

Dengan terraform, Anda dapat secara eksplisit mencantumkan satu file atau menggunakan wildcard * untuk mencantumkan setiap file dalam direktori.

> **Note**
>
> Wildcard hanya menyertakan file yang namanya diakhiri dengan `.tf`. File lain diabaikan.

```
infrastructure:
  templates:
    - file: "*"
      rendering_engine: hcl
      template_language: terraform
```

CodeBuildformat file manifes penyediaan berbasis:

Dengan penyediaan CodeBuild berbasis, Anda menentukan perintah shell penyediaan dan deprovisioning.

> **Note**
>
> Selain manifes, bundel Anda harus menyertakan file apa pun yang bergantung pada perintah Anda.

Contoh manifes berikut menggunakan penyediaan CodeBuild berbasis untuk sumber daya penyediaan (deploy) dan deprovision (destroy) menggunakan (). AWS Cloud Development Kit (AWS CDK) AWS CDK Bundel template juga harus menyertakan kode CDK.

Selama penyediaan, AWS Proton membuat file input dengan nilai untuk parameter input yang Anda tentukan dalam skema template dengan nama tersebut. `proton-input.json`

```
infrastructure:
  templates:
    - rendering_engine: codebuild
      settings:
        image: aws/codebuild/amazonlinux2-x86_64-standard:4.0
        runtimes:
          nodejs: 16
        provision:
          - npm install
          - npm run build
          - npm run cdk bootstrap
          - npm run cdk deploy -- --require-approval never --outputs-file proton-
outputs.json
          - jq 'to_entries | map_values(.value) | add | to_entries | map({key:.key,
 valueString:.value})' < proton-outputs.json > outputs.json
          - aws proton notify-resource-deployment-status-change --resource-arn
 $RESOURCE_ARN --status IN_PROGRESS --outputs file://./outputs.json
        deprovision:
          - npm install
          - npm run build
          - npm run cdk destroy
        project_properties:
          VpcConfig:
            VpcId: "{{ environment.inputs.codebuild_vpc_id }}"
            Subnets: "{{ environment.inputs.codebuild_subnets }}"
            SecurityGroupIds: "{{ environment.inputs.codebuild_security_groups }}"
```

Setelah menyiapkan direktori dan file manifes untuk paket template lingkungan atau layanan, Anda gzip direktori menjadi bola tar dan mengunggahnya ke bucket Amazon Simple Storage Service (Amazon S3) AWS Proton tempat dapat mengambilnya, atau ke repositori Git sinkronisasi template.

Saat Anda membuat versi minor lingkungan atau templat layanan yang Anda daftarkan AWS Proton, Anda menyediakan jalur ke lingkungan atau bola tar bundel templat layanan yang terletak di bucket S3 Anda. AWS Proton menyimpannya dengan versi minor template baru. Anda dapat memilih versi minor template baru untuk membuat atau memperbarui lingkungan atau layanan dengan AWS Proton.

Paket template lingkungan membungkus

Ada dua jenis bundel template lingkungan yang Anda buat AWS Proton.

- Untuk membuat bundel template lingkungan untuk template lingkungan standar, atur skema, infrastruktur sebagai file kode (IAc) dan file manifes dalam direktori seperti yang ditunjukkan dalam struktur direktori bundel template lingkungan berikut.
- Untuk membuat bundel template lingkungan untuk template lingkungan yang dikelola pelanggan, berikan hanya file skema dan direktori. Jangan sertakan direktori infrastruktur dan file. AWS Proton melempar kesalahan jika direktori infrastruktur dan file disertakan.

Untuk informasi selengkapnya, lihat Daftarkan dan terbitkan templat.

CloudFormation struktur direktori bundel template lingkungan:

```
/schema
  schema.yaml
/infrastructure
  manifest.yaml
  cloudformation.yaml
```

Struktur direktori bundel templat lingkungan Terraform:

```
/schema
  schema.yaml
/infrastructure
  manifest.yaml
  environment.tf
```

Paket template layanan membungkus

Untuk membuat paket template layanan, Anda harus mengatur file skema, infrastruktur sebagai kode (IAc), dan file manifes ke dalam direktori seperti yang ditunjukkan dalam contoh struktur direktori paket template layanan.

Jika Anda tidak menyertakan pipeline layanan dalam bundel template Anda, jangan sertakan direktori pipeline dan file dan atur `"pipelineProvisioning": "CUSTOMER_MANAGED"` saat Anda membuat template layanan yang akan dikaitkan dengan bundel template ini.

> **Note**
>
> Anda tidak dapat memodifikasi `pipelineProvisioning` setelah template layanan dibuat.

Untuk informasi selengkapnya, lihat Daftarkan dan terbitkan templat.

CloudFormation struktur direktori bundel template layanan:

```
/schema
  schema.yaml
/instance_infrastructure
  manifest.yaml
  cloudformation.yaml
/pipeline_infrastructure
  manifest.yaml
  cloudformation.yaml
```

Struktur direktori bundel template layanan Terraform:

```
/schema
  schema.yaml
/instance_infrastructure
  manifest.yaml
  instance.tf
/pipeline_infrastructure
  manifest.yaml
  pipeline.tf
```

Pertimbangan bundel template

- Infrastruktur sebagai file kode (IAc)

  AWS Proton audit template untuk format file yang benar. Namun, AWS Proton tidak memeriksa kesalahan pengembangan template, ketergantungan, dan logika. Misalnya, anggap Anda menentukan pembuatan bucket Amazon S3 di file CloudFormation IAC sebagai bagian dari templat layanan atau lingkungan Anda. Layanan dibuat berdasarkan template tersebut. Sekarang, misalkan pada titik tertentu Anda ingin menghapus layanan. Jika bucket S3 yang ditentukan tidak kosong dan file CloudFormation IAc tidak menandainya seperti `Retain` di`DeletionPolicy`, AWS Proton gagal pada operasi penghapusan layanan.
- Batas dan format ukuran file bundel
  - Ukuran file bundel, jumlah, dan batas ukuran nama dapat ditemukan diAWS Proton kuota.
  - Direktori bundel template file di-gzip ke dalam bola tar dan terletak di bucket Amazon Simple Storage Service (Amazon S3).
  - Setiap file dalam bundel harus berupa file YAML yang diformat valid.
- Enkripsi bundel template ember S3

  Jika Anda ingin mengenkripsi data sensitif dalam bundel template Anda saat istirahat di bucket S3 Anda, gunakan kunci SSE-S3 atau SSE-KMS untuk memungkinkan untuk mengambilnya. AWS Proton

AWS Proton template

Untuk menambahkan bundel AWS Proton template Anda ke pustaka template Anda, buat template versi minor dan daftarkan AWS Proton. Saat membuat template, berikan nama bucket Amazon S3 dan path untuk bundel template Anda. Setelah template diterbitkan, mereka dapat dipilih oleh anggota tim platform dan pengembang. Setelah dipilih, AWS Proton gunakan template untuk membuat dan menyediakan infrastruktur dan aplikasi.

Sebagai administrator, Anda dapat membuat dan mendaftarkan template lingkungan dengan AWS Proton. Template lingkungan ini kemudian dapat digunakan untuk menyebarkan beberapa lingkungan. Misalnya, dapat digunakan untuk menerapkan lingkungan “dev,” “staging,” dan “prod”. Lingkungan “dev” mungkin mencakup VPC dengan subnet pribadi dan kebijakan akses terbatas ke semua sumber daya. Output lingkungan dapat digunakan sebagai input untuk layanan.

Anda dapat membuat dan mendaftarkan template lingkungan untuk membuat dua jenis lingkungan yang berbeda. Baik Anda dan pengembang dapat menggunakan AWS Proton untuk menyebarkan layanan ke kedua jenis.

- Mendaftarkan dan mempublikasikan template lingkungan standar yang AWS Proton digunakan untuk menciptakan lingkungan standar yang menyediakan dan mengelola infrastruktur lingkungan.
- Daftarkan dan publikasikan templat lingkungan terkelola pelanggan yang AWS Proton digunakan untuk membuat lingkungan terkelola pelanggan yang terhubung ke infrastruktur yang telah disediakan. AWS Proton tidak mengelola infrastruktur penyediaan Anda yang ada.

Anda dapat membuat dan mendaftarkan templat layanan AWS Proton untuk menyebarkan layanan ke lingkungan. AWS Proton Lingkungan harus dibuat sebelum layanan dapat digunakan untuk itu.

Daftar berikut menjelaskan cara Anda membuat dan mengelola template dengan AWS Proton.

- (Opsional) Siapkan peran IAM untuk mengontrol akses pengembang ke panggilan AWS Proton API dan peran layanan AWS Proton IAM. Untuk informasi selengkapnya, lihat the section called “Peran IAM”.
- Tulis bundel template. Untuk informasi selengkapnya, lihat Bundel template.
- Buat dan daftarkan template AWS Proton setelah bundel template disusun, dikompresi, dan disimpan dalam bucket Amazon S3. Anda dapat melakukan ini baik di konsol atau dengan menggunakan AWS CLI.

- Uji dan gunakan templat untuk membuat dan mengelola sumber daya AWS Proton yang disediakan setelah terdaftar. AWS Proton
- Membuat dan mengelola versi utama dan minor dari template sepanjang umur template.

Anda dapat mengelola versi template secara manual atau dengan konfigurasi sinkronisasi template:

- Gunakan AWS Proton konsol dan AWS CLI untuk membuat versi minor atau mayor baru.
- Buat konfigurasi sinkronisasi templat yang memungkinkan AWS Proton secara otomatis membuat versi minor atau mayor baru saat mendeteksi perubahan pada bundel templat Anda di repositori yang Anda tentukan.

Untuk informasi tambahan, lihat Referensi API AWS Proton Layanan.

Topik

- Template berversi
- Daftarkan dan terbitkan templat
- Lihat data templat
- Perbarui template
- Hapus template
- Konfigurasi sinkronisasi templat
- Konfigurasi sinkronisasi layanan

Template berversi

Sebagai administrator atau anggota tim platform, Anda menentukan, membuat, dan mengelola pustaka templat berversi yang digunakan untuk menyediakan sumber daya infrastruktur. Ada dua jenis versi template — versi minor dan versi utama.

- Versi minor - Perubahan pada template yang memiliki skema kompatibel mundur. Perubahan ini tidak mengharuskan pengembang untuk memberikan informasi baru saat memperbarui ke versi template baru.

  Saat Anda mencoba membuat perubahan versi minor, AWS Proton lakukan upaya terbaik untuk menentukan apakah skema versi baru kompatibel dengan versi minor template sebelumnya. Jika skema baru tidak kompatibel ke belakang, AWS Proton gagal pendaftaran versi minor baru.

> **Note**
>
> Kompatibilitas ditentukan hanya berdasarkan skema. AWS Proton tidak memeriksa apakah file infrastruktur bundel templat sebagai kode (IAc) kompatibel dengan versi minor sebelumnya. Misalnya, AWS Proton tidak memeriksa apakah file IAC baru menyebabkan perubahan yang melanggar untuk aplikasi yang berjalan pada infrastruktur yang disediakan oleh versi minor template sebelumnya.

- Versi utama - Perubahan pada template yang mungkin tidak kompatibel ke belakang. Perubahan ini biasanya memerlukan masukan baru dari pengembang dan sering kali melibatkan perubahan skema template.

  Terkadang Anda dapat memilih untuk menetapkan perubahan yang kompatibel ke belakang sebagai versi utama berdasarkan model operasional tim Anda.

Cara AWS Proton menentukan apakah permintaan versi template untuk versi minor atau mayor tergantung pada cara perubahan template dilacak:

- Ketika Anda secara eksplisit membuat permintaan untuk membuat versi template baru, Anda meminta versi mayor dengan menentukan nomor versi mayor, dan Anda meminta versi minor dengan tidak menentukan nomor versi mayor.
- Saat Anda menggunakan sinkronisasi templat (dan karena itu Anda tidak membuat permintaan versi templat eksplisit), AWS Proton mencoba membuat versi minor baru untuk perubahan templat yang terjadi di file YAMAL yang ada. AWS Proton membuat versi mayor saat Anda membuat direktori baru untuk perubahan template baru (misalnya, pindah dari v1 ke v2).

> **Note**
>
> Registrasi versi minor baru berdasarkan sinkronisasi templat masih gagal jika AWS Proton menentukan bahwa perubahan tidak kompatibel ke belakang.

Ketika Anda mempublikasikan versi baru dari template, itu menjadi versi Rekomendasi jika itu adalah versi mayor dan minor tertinggi. AWS Proton Sumber daya baru dibuat menggunakan versi baru yang direkomendasikan, dan AWS Proton meminta administrator untuk menggunakan versi baru dan

memperbarui AWS Proton sumber daya yang ada yang menggunakan versi yang sudah ketinggalan zaman.

Daftarkan dan terbitkan templat

Anda dapat mendaftar dan mempublikasikan template lingkungan dan layanan dengan AWS Proton, seperti yang dijelaskan di bagian berikut.

Anda dapat membuat versi baru template dengan konsol atau AWS CLI.

Atau, Anda dapat menggunakan konsol atau AWS CLI membuat templat dan mengonfigurasi konfigurasi sinkronisasi templat untuknya. Konfigurasi ini memungkinkan AWS Proton sinkronisasi dari bundel template yang terletak di repositori git terdaftar yang telah Anda tentukan. Setiap kali komit didorong ke repositori Anda yang mengubah salah satu bundel template Anda, versi minor atau mayor baru dari template Anda akan dibuat, jika versi tersebut belum ada. Untuk mempelajari lebih lanjut tentang prasyarat dan persyaratan konfigurasi sinkronisasi templat, lihat. Konfigurasi sinkronisasi templat

Mendaftar dan mempublikasikan template lingkungan

Anda dapat mendaftar dan mempublikasikan jenis template lingkungan berikut.

- Mendaftarkan dan mempublikasikan template lingkungan standar yang AWS Proton digunakan untuk menyebarkan dan mengelola infrastruktur lingkungan.
- Daftarkan dan publikasikan templat lingkungan terkelola pelanggan yang AWS Proton digunakan untuk terhubung ke infrastruktur penyediaan yang sudah ada yang Anda kelola. AWS Proton tidak mengelola infrastruktur penyediaan Anda yang ada.

> **Important**
>
> Sebagai administrator, pastikan infrastruktur yang disediakan dan dikelola serta semua parameter keluaran kompatibel dengan templat lingkungan terkelola pelanggan terkait. AWS Proton tidak dapat memperhitungkan perubahan atas nama Anda karena perubahan ini tidak terlihat AWS Proton. Ketidakkonsistenan mengakibatkan kegagalan.

Anda dapat menggunakan konsol atau AWS CLI untuk mendaftar dan mempublikasikan template lingkungan.

Konsol Manajemen AWS

Gunakan konsol untuk mendaftar dan mempublikasikan template lingkungan baru.

1. Di AWS Proton konsol, pilih Template lingkungan.
2. Pilih Buat template lingkungan.
3. Di halaman Buat templat lingkungan, di bagian Opsi templat, pilih salah satu dari dua opsi templat yang tersedia.
   - Buat template untuk menyediakan lingkungan baru.
   - Buat template untuk menggunakan infrastruktur yang disediakan yang Anda kelola.
4. Jika Anda memilih Buat templat untuk menyediakan lingkungan baru, di bagian Sumber bundel Template, pilih salah satu dari tiga opsi sumber bundel templat yang tersedia. Untuk mempelajari lebih lanjut tentang persyaratan dan prasyarat untuk menyinkronkan templat, lihat. Konfigurasi sinkronisasi templat
   - Gunakan salah satu bundel template sampel kami.
   - Gunakan bundel template Anda sendiri.
   - Sinkronkan template dari Git.
5. Berikan jalur ke bundel template.
   a. Jika Anda memilih Gunakan salah satu bundel template sampel kami:

      Di bagian bundel templat sampel, pilih bundel templat sampel.

   b. Jika Anda memilih Sinkronkan template dari Git, di bagian Source code:

      i. Pilih repositori untuk konfigurasi sinkronisasi template Anda.

      ii. Masukkan nama cabang repositori untuk disinkronkan.

      iii. (Opsional) Masukkan nama direktori untuk membatasi pencarian bundel template Anda.

   c. Jika tidak, di bagian lokasi bundel S3, berikan jalur ke bundel template Anda.
6. Di bagian Detail Template.
   a. Masukkan nama Template.
   b. (Opsional) Masukkan nama tampilan Template.
   c. (Opsional) Masukkan deskripsi Template untuk template lingkungan.

7. (Opsional) Centang kotak centang untuk Sesuaikan pengaturan enkripsi (lanjutan) di bagian Pengaturan enkripsi untuk menyediakan kunci enkripsi Anda sendiri.
8. (Opsional) Di bagian Tag, pilih Tambahkan tag baru dan masukkan kunci dan nilai untuk membuat tag yang dikelola pelanggan.
9. Pilih template Create Environment.

   Anda sekarang berada di halaman baru yang menampilkan status dan detail untuk template lingkungan baru Anda. Rincian ini mencakup daftar AWS dan tag yang dikelola pelanggan. AWS Proton secara otomatis menghasilkan tag AWS terkelola untuk Anda saat Anda membuat AWS Proton sumber daya. Untuk informasi selengkapnya, lihat AWS Proton sumber daya dan penandaan.
10. Status status template lingkungan baru dimulai di status Draft. Anda dan orang lain dengan `proton:CreateEnvironment` izin dapat melihat dan mengaksesnya. Ikuti langkah selanjutnya untuk membuat template tersedia untuk orang lain.
11. Di bagian Versi Template, pilih tombol radio di sebelah kiri versi minor template yang baru saja Anda buat (1.0). Sebagai alternatif, Anda dapat memilih Publikasikan di peringatan info dan lewati langkah berikutnya.
12. Di bagian Versi templat, pilih Publikasikan.
13. Status template berubah menjadi Diterbitkan. Karena ini adalah versi terbaru dari template, itu adalah versi Rekomendasi.
14. Di panel navigasi, pilih Template lingkungan untuk melihat daftar templat dan detail lingkungan Anda.

Gunakan konsol untuk mendaftarkan versi mayor dan minor baru dari template lingkungan.

Untuk informasi selengkapnya, lihat Template berversi.

1. Di AWS Proton konsol, pilih Template Lingkungan.
2. Dalam daftar template lingkungan, pilih nama template lingkungan yang ingin Anda buat versi mayor atau minor.
3. Dalam tampilan detail template lingkungan, pilih Buat versi baru di bagian Versi template.
4. Di halaman Create a new environment template version, di bagian Sumber bundel Template, pilih salah satu dari dua opsi sumber bundel template yang tersedia.

   - Gunakan salah satu bundel template sampel kami.

- Gunakan bundel template Anda sendiri.

5. Berikan jalur ke bundel template yang dipilih.

   - Jika Anda memilih Gunakan salah satu bundel templat sampel kami, di bagian bundel templat sampel, pilih bundel templat sampel.
   - Jika Anda memilih Gunakan bundel templat Anda sendiri, di bagian lokasi bundel S3, pilih jalur ke bundel templat Anda.

6. Di bagian Detail Template.

   a. (Opsional) Masukkan nama tampilan Template.

   b. (Opsional) Masukkan deskripsi Template untuk template layanan.

7. Di bagian Detail templat, pilih salah satu opsi berikut.

   - Untuk membuat versi minor, biarkan kotak centang Centang untuk membuat versi mayor baru kosong.
   - Untuk membuat versi mayor, centang kotak Centang untuk membuat versi mayor baru.

8. Lanjutkan melalui langkah-langkah konsol untuk membuat versi minor atau mayor baru dan pilih Buat versi baru.

AWS CLI

Gunakan CLI untuk mendaftar dan mempublikasikan template lingkungan baru seperti yang ditunjukkan pada langkah-langkah berikut.

1. Buat templat lingkungan yang dikelola pelanggan OR standar dengan menentukan wilayah, nama, nama tampilan (opsional), dan deskripsi (opsional).

   a. Buat template lingkungan standar.

      Jalankan perintah berikut:

```
$ aws proton create-environment-template \
    --name "simple-env" \
    --display-name "Fargate" \
    --description "VPC with public access"
```

      Respons:

```
{
    "environmentTemplate": {
        "arn": "arn:aws:proton:region-id:123456789012:environment-template/
simple-env",
        "createdAt": "2020-11-11T23:02:45.336000+00:00",
        "description": "VPC with public access",
        "displayName": "VPC",
        "lastModifiedAt": "2020-11-11T23:02:45.336000+00:00",
        "name": "simple-env"
    }
}
```

b. Buat template lingkungan yang dikelola pelanggan dengan menambahkan `provisioning` parameter dengan nilai`CUSTOMER_MANAGED`.

Jalankan perintah berikut:

```
$ aws proton create-environment-template \
    --name "simple-env" \
    --display-name "Fargate" \
    --description "VPC with public access" \
    --provisioning "CUSTOMER_MANAGED"
```

Respons:

```
{
    "environmentTemplate": {
        "arn": "arn:aws:proton:region-id:123456789012:environment-template/
simple-env",
        "createdAt": "2020-11-11T23:02:45.336000+00:00",
        "description": "VPC with public access",
        "displayName": "VPC",
        "lastModifiedAt": "2020-11-11T23:02:45.336000+00:00",
        "name": "simple-env",
        "provisioning": "CUSTOMER_MANAGED"
    }
}
```

2. Buat versi minor 0 dari versi utama 1 dari template lingkungan

   Ini dan langkah-langkah yang tersisa sama untuk templat lingkungan standar dan yang dikelola pelanggan.

   Sertakan nama template, versi utama, dan nama bucket S3 serta kunci untuk bucket yang berisi bundel template lingkungan Anda.

   Jalankan perintah berikut:

```
$ aws proton create-environment-template-version \
    --template-name "simple-env" \
    --description "Version 1" \
    --source s3="{bucket=your_s3_bucket, key=your_s3_key}"
```

   Respons:

```
{
    "environmentTemplateVersion": {
        "arn": "arn:aws:proton:region-id:123456789012:environment-template/
simple-env:1.0",
        "createdAt": "2020-11-11T23:02:47.763000+00:00",
        "description": "Version 1",
        "lastModifiedAt": "2020-11-11T23:02:47.763000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "status": "REGISTRATION_IN_PROGRESS",
        "templateName": "simple-env"
    }
}
```

3. Gunakan perintah get untuk memeriksa status pendaftaran.

   Jalankan perintah berikut:

```
$ aws proton get-environment-template-version \
    --template-name "simple-env" \
    --major-version "1" \
    --minor-version "0"
```

   Respons:

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment-template/
simple-env:1.0",
        "createdAt": "2020-11-11T23:02:47.763000+00:00",
        "description": "Version 1",
        "lastModifiedAt": "2020-11-11T23:02:47.763000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "recommendedMinorVersion": "0",
        "schema": "schema:\n  format:\n    openapi: \"3.0.0\"\n
 environment_input_type: \"MyEnvironmentInputType\"\n  types:\n
 MyEnvironmentInputType:\n      type: object\n      description: \"Input
 properties for my environment\"\n      properties:\n        my_sample_input:\n
          type: string\n          description: \"This is a sample input\"\n
      default: \"hello world\"\n        my_other_sample_input:\n          type:
 string\n          description: \"Another sample input\"\n      required:\n
   - my_other_sample_input\n",
        "status": "DRAFT",
        "statusMessage": "",
        "templateName": "simple-env"
    }
}
```

4. Publikasikan versi minor 0 dari mayor versi 1 dari template lingkungan dengan memberikan nama template dan versi mayor dan minor. Versi ini adalah `Recommended` versinya.

   Jalankan perintah berikut:

```
$ aws proton update-environment-template-version \
    --template-name "simple-env" \
    --major-version "1" \
    --minor-version "0" \
    --status "PUBLISHED"
```

   Respons:

```
{
    "environmentTemplateVersion": {
        "arn": "arn:aws:proton:region-id:123456789012:environment-template/
simple-env:1.0",
```

```
        "createdAt": "2020-11-11T23:02:47.763000+00:00",
        "description": "Version 1",
        "lastModifiedAt": "2020-11-11T23:02:54.610000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "recommendedMinorVersion": "0",
        "schema": "schema:\n  format:\n    openapi: \"3.0.0\"\n
 environment_input_type: \"MyEnvironmentInputType\"\n  types:\n
 MyEnvironmentInputType:\n      type: object\n      description: \"Input
 properties for my environment\"\n      properties:\n        my_sample_input:\n
          type: string\n          description: \"This is a sample input\"\n
      default: \"hello world\"\n        my_other_sample_input:\n          type:
 string\n          description: \"Another sample input\"\n      required:\n
   - my_other_sample_input\n",
        "status": "PUBLISHED",
        "statusMessage": "",
        "templateName": "simple-env"
    }
}
```


Setelah membuat template baru menggunakan AWS CLI, Anda dapat melihat daftar AWS dan tag yang dikelola pelanggan. AWS Proton secara otomatis menghasilkan tag AWS terkelola untuk Anda. Anda juga dapat memodifikasi dan membuat tag yang dikelola pelanggan menggunakan AWS CLI. Untuk informasi selengkapnya, lihat AWS Proton sumber daya dan penandaan.

Jalankan perintah berikut:

```
$ aws proton list-tags-for-resource \
    --resource-arn "arn:aws:proton:region-id:123456789012:environment-
template/simple-env"
```

Mendaftar dan mempublikasikan template layanan

Saat membuat versi templat layanan, Anda menentukan daftar templat lingkungan yang kompatibel. Dengan begitu, ketika pengembang memilih template layanan, mereka memiliki opsi untuk lingkungan mana untuk menyebarkan layanan mereka.

Sebelum membuat layanan dari templat layanan atau sebelum memublikasikan templat layanan, konfirmasikan bahwa lingkungan digunakan dari templat lingkungan kompatibel yang terdaftar.

Anda tidak dapat memperbarui layanan ke versi utama baru jika digunakan ke lingkungan yang dibangun dari templat lingkungan yang kompatibel yang dihapus.

Untuk menambah atau menghapus templat lingkungan yang kompatibel untuk versi templat layanan, Anda membuat versi utama baru.

Anda dapat menggunakan konsol atau AWS CLI untuk mendaftar dan mempublikasikan template layanan.

Konsol Manajemen AWS

Gunakan konsol untuk mendaftar dan mempublikasikan template layanan baru.

1. Di AWS Proton konsol, pilih Templat layanan.
2. Pilih Buat template layanan.
3. Di halaman Buat templat layanan, di bagian Sumber bundel Template, pilih salah satu opsi templat yang tersedia.
   - Gunakan bundel template Anda sendiri.
   - Sinkronkan template dari Git.
4. Berikan jalur ke bundel template.
   a. Jika Anda memilih Sinkronkan template dari Git, di bagian repositori kode Sumber:
      i. Pilih repositori untuk konfigurasi sinkronisasi template Anda.
      ii. Masukkan nama cabang repositori untuk disinkronkan.
      iii. (Opsional) Masukkan nama direktori untuk membatasi pencarian bundel template Anda.
   b. Jika tidak, di bagian lokasi bundel S3, berikan jalur ke bundel template Anda.
5. Di bagian Detail Template.
   a. Masukkan nama Template.
   b. (Opsional) Masukkan nama tampilan Template.
   c. (Opsional) Masukkan deskripsi Template untuk template layanan.
6. Di bagian Template lingkungan yang kompatibel, pilih dari daftar templat lingkungan yang kompatibel.

7. (Opsional) Di bagian Pengaturan enkripsi, pilih Sesuaikan pengaturan enkripsi (lanjutan) untuk menyediakan kunci enkripsi Anda sendiri.
8. (Opsional) Di bagian Pipeline:

   Jika Anda tidak menyertakan definisi saluran layanan dalam templat layanan, hapus centang pada kotak centang Pipeline - opsional di bagian bawah halaman. Anda tidak dapat mengubah ini setelah template layanan dibuat. Untuk informasi selengkapnya, lihat Bundel template.
9. (Opsional) Di bagian Sumber komponen yang didukung, untuk sumber Komponen, pilih Ditentukan secara langsung untuk mengaktifkan lampiran komponen yang ditentukan secara langsung ke instance layanan Anda.
10. (Opsional) Di bagian Tag, pilih Tambahkan tag baru dan masukkan kunci dan nilai untuk membuat tag yang dikelola pelanggan.
11. Pilih Buat templat layanan.

    Anda sekarang berada di halaman baru yang menampilkan status dan detail untuk template layanan baru Anda. Rincian ini mencakup daftar AWS dan tag yang dikelola pelanggan. AWS Proton secara otomatis menghasilkan tag AWS terkelola untuk Anda saat Anda membuat AWS Proton sumber daya. Untuk informasi selengkapnya, lihat AWS Proton sumber daya dan penandaan.
12. Status status template layanan baru dimulai di status Draft. Anda dan orang lain dengan `proton:CreateService` izin dapat melihat dan mengaksesnya. Ikuti langkah selanjutnya untuk membuat template tersedia untuk orang lain.
13. Di bagian Versi Template, pilih tombol radio di sebelah kiri versi minor template yang baru saja Anda buat (1.0). Sebagai alternatif, Anda dapat memilih Publikasikan di peringatan info dan lewati langkah berikutnya.
14. Di bagian Versi templat, pilih Publikasikan.
15. Status template berubah menjadi Diterbitkan. Karena ini adalah versi terbaru dari template, itu adalah versi Rekomendasi.
16. Di panel navigasi, pilih Templat layanan untuk melihat daftar templat dan detail layanan Anda.

Gunakan konsol untuk mendaftarkan versi mayor dan minor baru dari template layanan.

Untuk informasi selengkapnya, lihat Template berversi.

1. Di AWS Proton konsol, pilih Templat Layanan.

2. Dalam daftar templat layanan, pilih nama templat layanan yang ingin Anda buat versi mayor atau minor.
3. Dalam tampilan detail template layanan, pilih Buat versi baru di bagian Versi template.
4. Di halaman Create a new service template version, di bagian Bundle source, pilih Use your own template bundle.
5. Di bagian lokasi bundel S3, pilih jalur ke bundel template Anda.
6. Di bagian Detail Template.

   a. (Opsional) Masukkan nama tampilan Template.

   b. (Opsional) Masukkan deskripsi Template untuk template layanan.
7. Di bagian Detail templat, pilih salah satu opsi berikut.

   - Untuk membuat versi minor, biarkan kotak centang Centang untuk membuat versi mayor baru kosong.
   - Untuk membuat versi mayor, centang kotak Centang untuk membuat versi mayor baru.
8. Lanjutkan melalui langkah-langkah konsol untuk membuat versi minor atau mayor baru dan pilih Buat versi baru.

AWS CLI

Untuk membuat template layanan yang menyebarkan layanan tanpa pipeline layanan, tambahkan parameter dan nilai `--pipeline-provisioning "CUSTOMER_MANAGED"` ke `create-service-template` perintah. Konfigurasikan bundel template Anda seperti yang dijelaskan dalam Bundel template pembuatan danPersyaratan skema untuk bundel templat layanan.

> **Note**
>
> Anda tidak dapat memodifikasi `pipelineProvisioning` setelah template layanan dibuat.

1. Gunakan CLI untuk mendaftar dan mempublikasikan template layanan baru, dengan atau tanpa pipeline layanan, seperti yang ditunjukkan pada langkah-langkah berikut.

   a. Buat template layanan dengan pipeline layanan menggunakan CLI.

      Tentukan nama, nama tampilan (opsional), dan deskripsi (opsional).

Jalankan perintah berikut:

```
$ aws proton create-service-template \
    --name "fargate-service" \
    --display-name "Fargate" \
    --description "Fargate-based Service"
```

Respons:

```
{
    "serviceTemplate": {
        "arn": "arn:aws:proton:region-id:123456789012:service-template/
fargate-service",
        "createdAt": "2020-11-11T23:02:55.551000+00:00",
        "description": "Fargate-based Service",
        "displayName": "Fargate",
        "lastModifiedAt": "2020-11-11T23:02:55.551000+00:00",
        "name": "fargate-service"
    }
}
```

b. Buat template layanan tanpa pipeline layanan.

Tambahkan `--pipeline-provisioning`.

Jalankan perintah berikut:

```
$ aws proton create-service-template \
    --name "fargate-service" \
    --display-name "Fargate" \
    --description "Fargate-based Service" \
    --pipeline-provisioning "CUSTOMER_MANAGED"
```

Respons:

```
{
    "serviceTemplate": {
        "arn": "arn:aws:proton:region-id:123456789012:service-template/
fargate-service",
        "createdAt": "2020-11-11T23:02:55.551000+00:00",
        "description": "Fargate-based Service",
```

```
        "displayName": "Fargate",
        "lastModifiedAt": "2020-11-11T23:02:55.551000+00:00",
        "name": "fargate-service",
        "pipelineProvisioning": "CUSTOMER_MANAGED"
    }
}
```

2. Buat versi minor 0 dari versi utama 1 dari template layanan.

   Sertakan nama template, template lingkungan yang kompatibel, versi utama, serta nama bucket S3 serta kunci untuk bucket yang berisi paket template layanan Anda.

   Jalankan perintah berikut:

```
$ aws proton create-service-template-version \
    --template-name "fargate-service" \
    --description "Version 1" \
    --source s3="{bucket=your_s3_bucket, key=your_s3_key}" \
    --compatible-environment-templates '[{"templateName":"simple-
env","majorVersion":"1"}]'
```

   Respons:

```
{
    "serviceTemplateMinorVersion": {
        "arn": "arn:aws:proton:region-id:123456789012:service-template/fargate-
service:1.0",
        "compatibleEnvironmentTemplates": [
            {
                "majorVersion": "1",
                "templateName": "simple-env"
            }
        ],
        "createdAt": "2020-11-11T23:02:57.912000+00:00",
        "description": "Version 1",
        "lastModifiedAt": "2020-11-11T23:02:57.912000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "status": "REGISTRATION_IN_PROGRESS",
        "templateName": "fargate-service"
    }
}
```

3. Gunakan perintah get untuk memeriksa status pendaftaran.

   Jalankan perintah berikut:

```
$ aws proton get-service-template-version \
    --template-name "fargate-service" \
    --major-version "1" \
    --minor-version "0"
```

   Respons:

```
{
    "serviceTemplateMinorVersion": {
        "arn": "arn:aws:proton:us-east-1:123456789012:service-template/fargate-
service:1.0",
        "compatibleEnvironmentTemplates": [
            {
                "majorVersion": "1",
                "templateName": "simple-env"
            }
        ],
        "createdAt": "2020-11-11T23:02:57.912000+00:00",
        "description": "Version 1",
        "lastModifiedAt": "2020-11-11T23:02:57.912000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "schema": "schema:\n  format:\n    openapi: \"3.0.0\"\n
 pipeline_input_type: \"MyPipelineInputType\"\n  service_input_type:
 \"MyServiceInstanceInputType\"\n\n  types:\n    MyPipelineInputType:\n
    type: object\n      description: \"Pipeline input properties\"\n
 required:\n        - my_sample_pipeline_required_input\n      properties:\n
        my_sample_pipeline_optional_input:\n          type: string\n
   description: \"This is a sample input\"\n          default: \"hello world
\"\n        my_sample_pipeline_required_input:\n          type: string\n
     description: \"Another sample input\"\n\n    MyServiceInstanceInputType:
\n      type: object\n      description: \"Service instance input properties
\"\n      required:\n        - my_sample_service_instance_required_input\n
    properties:\n        my_sample_service_instance_optional_input:\n
   type: string\n          description: \"This is a sample input\"\n
 default: \"hello world\"\n        my_sample_service_instance_required_input:\n
         type: string\n          description: \"Another sample input\"",
        "status": "DRAFT",
```

```
        "statusMessage": "",
        "templateName": "fargate-service"
    }
}
```

4. Publikasikan template layanan dengan menggunakan perintah update untuk mengubah status menjadi"PUBLISHED".

   Jalankan perintah berikut:

```
$ aws proton update-service-template-version \
    --template-name "fargate-service" \
    --description "Version 1" \
    --major-version "1" \
    --minor-version "0" \
    --status "PUBLISHED"
```

   Respons:

```
{
    "serviceTemplateVersion": {
        "arn": "arn:aws:proton:region-id:123456789012:service-template/fargate-
service:1.0",
        "compatibleEnvironmentTemplates": [
            {
                "majorVersion": "1",
                "templateName": "simple-env"
            }
        ],
        "createdAt": "2020-11-11T23:02:57.912000+00:00",
        "description": "Version 1",
        "lastModifiedAt": "2020-11-11T23:02:57.912000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "recommendedMinorVersion": "0",
        "schema": "schema:\n  format:\n    openapi: \"3.0.0\"\n
 pipeline_input_type: \"MyPipelineInputType\"\n  service_input_type:
 \"MyServiceInstanceInputType\"\n\n  types:\n    MyPipelineInputType:\n
    type: object\n      description: \"Pipeline input properties\"\n
 required:\n        - my_sample_pipeline_required_input\n      properties:\n
        my_sample_pipeline_optional_input:\n          type: string\n
 description: \"This is a sample input\"\n          default: \"hello pipeline
\"\n        my_sample_pipeline_required_input:\n          type: string\n
```

```
     description: \"Another sample input\"\n\n    MyServiceInstanceInputType:
\n      type: object\n      description: \"Service instance input properties
\"\n      required:\n        - my_sample_service_instance_required_input\n
    properties:\n        my_sample_service_instance_optional_input:\n
   type: string\n          description: \"This is a sample input\"\n
 default: \"hello world\"\n        my_sample_service_instance_required_input:\n
         type: string\n          description: \"Another sample input\"\n",
        "status": "PUBLISHED",
        "statusMessage": "",
        "templateName": "fargate-service"
    }
}
```

5. Periksa apakah AWS Proton telah menerbitkan versi 1.0 dengan menggunakan perintah get untuk mengambil data detail template layanan.

   Jalankan perintah berikut:

```
$ aws proton get-service-template-version \
    --template-name "fargate-service" \
    --major-version "1" \
    --minor-version "0"
```

   Respons:

```
{
    "serviceTemplateMinorVersion": {
        "arn": "arn:aws:proton:us-east-1:123456789012:service-template/fargate-
service:1.0",
        "compatibleEnvironmentTemplates": [
            {
                "majorVersion": "1",
                "templateName": "simple-env"
            }
        ],
        "createdAt": "2020-11-11T23:02:57.912000+00:00",
        "description": "Version 1",
        "lastModifiedAt": "2020-11-11T23:03:04.767000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "schema": "schema:\n  format:\n    openapi: \"3.0.0\"\n
 pipeline_input_type: \"MyPipelineInputType\"\n  service_input_type:
 \"MyServiceInstanceInputType\"\n\n  types:\n    MyPipelineInputType:\n
```

```
    type: object\n      description: \"Pipeline input properties\"\n
 required:\n        - my_sample_pipeline_required_input\n      properties:\n
        my_sample_pipeline_optional_input:\n          type: string\n
   description: \"This is a sample input\"\n          default: \"hello world
\"\n        my_sample_pipeline_required_input:\n          type: string\n
     description: \"Another sample input\"\n\n    MyServiceInstanceInputType:
\n      type: object\n      description: \"Service instance input properties
\"\n      required:\n        - my_sample_service_instance_required_input\n
    properties:\n        my_sample_service_instance_optional_input:\n
   type: string\n          description: \"This is a sample input\"\n
 default: \"hello world\"\n        my_sample_service_instance_required_input:\n
         type: string\n          description: \"Another sample input\"",
        "status": "PUBLISHED",
        "statusMessage": "",
        "templateName": "fargate-service"
    }
}
```

Lihat data templat

Anda dapat melihat daftar templat dengan detail dan melihat templat individual dengan data detail dengan menggunakan AWS Proton konsol dan AWS CLI.

Data template lingkungan yang dikelola pelanggan mencakup `provisioned` parameter dengan nilai`CUSTOMER_MANAGED`.

Jika template layanan tidak menyertakan pipeline layanan, data template layanan menyertakan `pipelineProvisioning` parameter dengan nilai`CUSTOMER_MANAGED`.

Untuk informasi selengkapnya, lihat Daftarkan dan terbitkan templat.

Anda dapat menggunakan konsol atau AWS CLI untuk daftar dan melihat data template.

Konsol Manajemen AWS

Gunakan konsol untuk membuat daftar dan melihat templat.

1. Untuk melihat daftar templat, pilih templat (Lingkungan atau Layanan).
2. Untuk melihat data detail pilih nama template.

Lihat data detail template, daftar versi mayor dan minor template, daftar AWS Proton sumber daya yang digunakan menggunakan versi template dan tag template.

Versi utama dan versi minor yang direkomendasikan diberi label sebagai Direkomendasikan.

AWS CLI

Gunakan AWS CLI untuk daftar dan lihat templat.

Jalankan perintah berikut:

```
$ aws proton get-environment-template-version \
    --template-name "simple-env" \
    --major-version "1" \
    --minor-version "0"
```

Respons:

```
{
    "environmentTemplateVersion": {
        "arn": "arn:aws:proton:region-id:123456789012:environment-template/simple-
env:1.0",
        "createdAt": "2020-11-10T18:35:08.293000+00:00",
        "description": "Version 1",
        "lastModifiedAt": "2020-11-10T18:35:11.162000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "recommendedMinorVersion": "0",
        "schema": "schema:\n  format:\n    openapi: \"3.0.0\"\n
 environment_input_type: \"MyEnvironmentInputType\"\n  types:\n
 MyEnvironmentInputType:\n      type: object\n      description: \"Input properties
 for my environment\"\n      properties:\n        my_sample_input:\n
  type: string\n          description: \"This is a sample input\"\n
 default: \"hello world\"\n        my_other_sample_input:\n          type: string
\n          description: \"Another sample input\"\n      required:\n        -
 my_other_sample_input\n",
        "status": "DRAFT",
        "statusMessage": "",
        "templateName": "simple-env"
    }
}
```

Jalankan perintah berikut:

```
$ aws proton list-environment-templates
```

Respons:

```
{
    "templates": [
        {
            "arn": "arn:aws:proton:region-id:123456789012:environment-template/
simple-env-3",
            "createdAt": "2020-11-10T18:35:05.763000+00:00",
            "description": "VPC with Public Access",
            "displayName": "VPC",
            "lastModifiedAt": "2020-11-10T18:35:05.763000+00:00",
            "name": "simple-env-3",
            "recommendedVersion": "1.0"
        },
        {
            "arn": "arn:aws:proton:region-id:123456789012:environment-template/
simple-env-1",
            "createdAt": "2020-11-10T00:14:06.881000+00:00",
            "description": "Some SSM Parameters",
            "displayName": "simple-env-1",
            "lastModifiedAt": "2020-11-10T00:14:06.881000+00:00",
            "name": "simple-env-1",
            "recommendedVersion": "1.0"
        }
    ]
}
```

Lihat versi minor dari template layanan.

Jalankan perintah berikut:

```
$ aws proton get-service-template-version \
    --template-name "fargate-service" \
    --major-version "1" \
    --minor-version "0"
```

Respons:

```
{
    "serviceTemplateMinorVersion": {
        "arn": "arn:aws:proton:us-east-1:123456789012:service-template/fargate-
service:1.0",
        "compatibleEnvironmentTemplates": [
            {
                "majorVersion": "1",
                "templateName": "simple-env"
            }
        ],
        "createdAt": "2020-11-11T23:02:57.912000+00:00",
        "description": "Version 1",
        "lastModifiedAt": "2020-11-11T23:02:57.912000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "schema": "schema:\n  format:\n    openapi: \"3.0.0\"\n
 pipeline_input_type: \"MyPipelineInputType\"\n  service_input_type:
 \"MyServiceInstanceInputType\"\n\n  types:\n    MyPipelineInputType:\n
    type: object\n      description: \"Pipeline input properties\"\n
 required:\n        - my_sample_pipeline_required_input\n      properties:\n
        my_sample_pipeline_optional_input:\n          type: string\n
 description: \"This is a sample input\"\n          default: \"hello world\"\n
  my_sample_pipeline_required_input:\n          type: string\n          description:
 \"Another sample input\"\n\n    MyServiceInstanceInputType:\n      type: object
\n      description: \"Service instance input properties\"\n      required:\n
       - my_sample_service_instance_required_input\n      properties:\n
 my_sample_service_instance_optional_input:\n          type: string\n
 description: \"This is a sample input\"\n          default: \"hello world\"\n
     my_sample_service_instance_required_input:\n          type: string\n
 description: \"Another sample input\"",
        "status": "DRAFT",
        "statusMessage": "",
        "templateName": "fargate-service"
    }
}
```


Lihat template layanan tanpa pipeline layanan seperti yang ditunjukkan pada contoh perintah dan respons berikutnya.

Jalankan perintah berikut:

```
$ aws proton get-service-template \
```

```
    --name "simple-svc-template-cli"
```

Respons:

```
{
    "serviceTemplate": {
        "arn": "arn:aws:proton:region-id:123456789012:service-template/simple-svc-
template-cli",
        "createdAt": "2021-02-18T15:38:57.949000+00:00",
        "displayName": "simple-svc-template-cli",
        "lastModifiedAt": "2021-02-18T15:38:57.949000+00:00",
        "status": "DRAFT",
        "name": "simple-svc-template-cli",
        "pipelineProvisioning": "CUSTOMER_MANAGED"
    }
}
```

Perbarui template

Anda dapat memperbarui template seperti yang dijelaskan dalam daftar berikut.

- Edit `description` atau `display name` templat saat Anda menggunakan konsol atau AWS CLI. Anda tidak dapat mengedit template. `name`
- Perbarui status template versi minor saat Anda menggunakan konsol atau AWS CLI. Anda hanya dapat mengubah status dari `DRAFT` ke`PUBLISHED`.
- Edit nama tampilan dan deskripsi template versi minor atau utama saat Anda menggunakan template AWS CLI.

Konsol Manajemen AWS

Edit deskripsi templat dan nama tampilan menggunakan konsol seperti yang dijelaskan dalam langkah-langkah berikut.

Dalam daftar template.

1. Di AWS Proton konsol, pilih Template (Lingkungan atau Layanan).
2. Dalam daftar templat, pilih tombol radio di sebelah kiri templat yang ingin Anda perbarui deskripsi atau nama tampilannya.

3. Pilih Tindakan dan kemudian Edit.
4. Di halaman templat Edit (lingkungan atau layanan), di bagian Detail templat, masukkan hasil edit Anda di formulir dan pilih Simpan perubahan.

Ubah status versi minor template menggunakan konsol untuk mempublikasikan template seperti yang dijelaskan di bawah ini. Anda hanya dapat mengubah status dari `DRAFT` ke`PUBLISHED`.

Di halaman detail template (lingkungan atau layanan).

1. Di AWS Proton konsol, pilih templat (Lingkungan atau Layanan).
2. Dalam daftar templat, pilih nama templat yang ingin Anda perbarui status versi minor dari Draft ke Published.
3. Di halaman detail templat (lingkungan atau layanan), di bagian Versi templat, pilih tombol radio di sebelah kiri versi minor yang ingin Anda terbitkan.
4. Pilih Publikasikan di bagian Versi templat. Status berubah dari Draft ke Published.

AWS CLI

Contoh perintah dan respons berikut menunjukkan bagaimana Anda dapat mengedit deskripsi template lingkungan.

Jalankan perintah berikut.

```
$ aws proton update-environment-template \
    --name "simple-env" \
    --description "A single VPC with public access"
```

Respons:

```
{
    "environmentTemplate": {
        "arn": "arn:aws:proton:region-id:123456789012:environment-template/simple-
env",
        "createdAt": "2020-11-28T22:02:10.651000+00:00",
        "description": "A single VPC with public access",
        "displayName": "simple-env",
        "lastModifiedAt": "2020-11-29T16:11:18.956000+00:00",
        "majorVersion": "1",
```

```
        "minorVersion": "0",
        "recommendedMinorVersion": "0",
        "schema": "schema:\n  format:\n    openapi: \"3.0.0\"\n
 environment_input_type: \"MyEnvironmentInputType\"\n  types:\n
 MyEnvironmentInputType:\n      type: object\n      description: \"Input properties
 for my environment\"\n      properties:\n        my_sample_input:\n
  type: string\n          description: \"This is a sample input\"\n
 default: \"hello world\"\n        my_other_sample_input:\n          type: string
\n          description: \"Another sample input\"\n      required:\n        -
 my_other_sample_input\n",
        "status": "PUBLISHED",
        "statusMessage": "",
        "templateName": "simple-env"
    }
}
```

Anda juga dapat menggunakan AWS CLI untuk memperbarui template layanan. LihatMendaftar dan mempublikasikan template layanan, langkah 5, untuk contoh memperbarui status versi minor dari template layanan.

Hapus template

Template dapat dihapus menggunakan konsol dan AWS CLI.

Anda dapat menghapus versi minor template lingkungan jika tidak ada lingkungan yang diterapkan ke versi tersebut.

Anda dapat menghapus versi minor template layanan jika tidak ada instance layanan atau pipeline yang disebarkan ke versi tersebut. Pipeline Anda dapat diterapkan ke versi template yang berbeda dari instance layanan Anda. Misalnya, jika instance layanan Anda diperbarui ke versi 1.1 dari 1.0 dan pipeline Anda masih di-deploy ke versi 1.0, Anda tidak dapat menghapus template layanan 1.0.

Konsol Manajemen AWS

Anda dapat menggunakan konsol untuk menghapus seluruh template atau versi minor dan mayor individual dari template.

Gunakan konsol untuk menghapus template sebagai berikut.

> **Note**
>
> Saat menggunakan konsol untuk menghapus templat.
>
> - Ketika Anda menghapus seluruh template, Anda juga menghapus versi mayor dan minor dari template.

Dalam daftar templat (lingkungan atau layanan).

1. Di AWS Proton konsol, pilih Template (Lingkungan atau Layanan).
2. Dalam daftar templat, pilih tombol radio di sebelah kiri templat yang ingin Anda hapus.

   Anda hanya dapat menghapus seluruh template jika tidak ada AWS Proton sumber daya yang digunakan ke versinya.
3. Pilih Tindakan dan kemudian Hapus untuk menghapus seluruh template.
4. Modal meminta Anda untuk mengonfirmasi tindakan hapus.
5. Ikuti instruksi dan pilih Ya, hapus.

Di halaman detail template (lingkungan atau layanan).

1. Di AWS Proton konsol, pilih Template (Lingkungan atau Layanan).
2. Dalam daftar templat, pilih nama templat yang ingin Anda hapus seluruhnya atau hapus versi mayor atau minor individual darinya.
3. Untuk menghapus seluruh template.

   Anda hanya dapat menghapus seluruh template jika tidak ada AWS Proton sumber daya yang digunakan ke versinya.

   a. Pilih Hapus, sudut kanan atas halaman.

   b. Modal meminta Anda untuk mengonfirmasi tindakan hapus.

   c. Ikuti instruksi dan pilih Ya, hapus.
4. Untuk menghapus versi mayor atau minor dari template.

   Anda hanya dapat menghapus versi minor template jika tidak ada AWS Proton sumber daya yang diterapkan ke versi tersebut.

a. Di bagian Versi templat, pilih tombol radio di sebelah kiri versi yang ingin Anda hapus.
b. Pilih Hapus di bagian Versi templat.
c. Modal meminta Anda untuk mengonfirmasi tindakan hapus.
d. Ikuti instruksi dan pilih Ya, hapus.

AWS CLI

AWS CLI operasi penghapusan template tidak termasuk penghapusan versi lain dari template. Saat menggunakan AWS CLI, hapus templat dengan kondisi berikut.

- Hapus seluruh template jika tidak ada versi minor atau utama dari template yang ada.
- Hapus versi mayor saat Anda menghapus versi minor terakhir yang tersisa.
- Hapus versi minor template jika tidak ada AWS Proton sumber daya yang diterapkan ke versi tersebut.
- Hapus versi minor template yang direkomendasikan jika tidak ada versi minor lain dari template yang ada dan tidak ada AWS Proton sumber daya yang digunakan untuk versi tersebut.

Contoh perintah dan tanggapan berikut menunjukkan cara menggunakan AWS CLI untuk menghapus template.

Jalankan perintah berikut:

```
$ aws proton delete-environment-template-version \
    --template-name "simple-env" \
    --major-version "1" \
    --minor-version "0"
```

Respons:

```
{
    "environmentTemplateVersion": {
        "arn": "arn:aws:proton:region-id:123456789012:environment-template/simple-
env:1.0",
        "createdAt": "2020-11-11T23:02:47.763000+00:00",
        "description": "Version 1",
        "lastModifiedAt": "2020-11-11T23:02:54.610000+00:00",
        "majorVersion": "1",
```

```
        "minorVersion": "0",
        "status": "PUBLISHED",
        "statusMessage": "",
        "templateName": "simple-env"
    }
}
```

Jalankan perintah berikut:

```
$ aws proton delete-environment-template \
    --name "simple-env"
```

Respons:

```
{
    "environmentTemplate": {
        "arn": "arn:aws:proton:region-id:123456789012:environment-template/simple-
env",
        "createdAt": "2020-11-11T23:02:45.336000+00:00",
        "description": "VPC with Public Access",
        "displayName": "VPC",
        "lastModifiedAt": "2020-11-12T00:23:22.339000+00:00",
        "name": "simple-env",
        "recommendedVersion": "1.0"
    }
}
```

Jalankan perintah berikut:

```
$ aws proton delete-service-template-version \
    --template-name "fargate-service" \
    --major-version "1" \
    --minor-version "0"
```

Respons:

```
{
    "serviceTemplateVersion": {
        "arn": "arn:aws:proton:region-id:123456789012:service-template/fargate-
service:1.0",
        "compatibleEnvironmentTemplates": [{"majorVersion": "1", "templateName":
 "simple-env"}],
```

```
        "createdAt": "2020-11-28T22:07:05.798000+00:00",
        "lastModifiedAt": "2020-11-28T22:19:05.368000+00:00",
        "majorVersion": "1",
        "minorVersion": "0",
        "status": "PUBLISHED",
        "statusMessage": "",
        "templateName": "fargate-service"
    }
}
```

Konfigurasi sinkronisasi templat

Pelajari cara mengonfigurasi templat agar memungkinkan AWS Proton sinkronisasi dari bundel templat yang terletak di repositori git terdaftar yang Anda tentukan. Saat komit didorong ke repositori Anda, AWS Proton periksa perubahan pada bundel template repositori Anda. Jika mendeteksi perubahan bundel template, versi minor atau mayor baru dari templatnya akan dibuat, jika versinya belum ada. AWS Proton saat ini mendukung GitHub, GitHub Enterprise, dan BitBucket.

Mendorong komit ke bundel template yang disinkronkan

Saat Anda mendorong komit ke cabang yang dilacak oleh salah satu templat Anda, AWS Proton mengkloning repositori Anda dan menentukan templat apa yang perlu disinkronkan. Ini memindai file di direktori Anda untuk menemukan direktori yang cocok dengan konvensi. `{template-name}/{major-version}/`

Setelah AWS Proton menentukan template dan versi utama mana yang terkait dengan repositori dan cabang Anda, ia mulai mencoba menyinkronkan semua template tersebut secara paralel.

Selama setiap sinkronisasi ke template tertentu, periksa AWS Proton terlebih dahulu untuk melihat apakah konten direktori template berubah sejak sinkronisasi terakhir berhasil. Jika konten tidak berubah, AWS Proton lewati mendaftarkan bundel duplikat. Ini memastikan bahwa versi minor template baru dibuat jika konten bundel template berubah. Jika isi bundel template berubah, bundel terdaftar dengan AWS Proton.

Setelah bundel templat terdaftar, AWS Proton pantau status pendaftaran hingga pendaftaran selesai.

Hanya satu sinkronisasi yang dapat terjadi pada template tertentu versi minor dan mayor pada satu waktu tertentu. Komit apa pun yang mungkin telah didorong saat sinkronisasi sedang berlangsung akan dikelompokkan. Komit batch disinkronkan setelah upaya sinkronisasi sebelumnya selesai.

Menyinkronkan templat layanan

AWS Proton dapat menyinkronkan template lingkungan dan layanan dari repositori git Anda. Untuk menyinkronkan templat layanan Anda, Anda menambahkan file tambahan bernama `.template-registration.yaml` ke setiap direktori versi utama dalam bundel templat Anda. File ini berisi detail tambahan yang AWS Proton diperlukan saat membuat versi templat layanan untuk Anda mengikuti komit: lingkungan yang kompatibel dan sumber komponen yang didukung.

Jalur lengkap file adalah*`service-template-name`*`/`*`major-version`*`/.template-registration.yaml`. Untuk informasi selengkapnya, lihat the section called “Menyinkronkan templat layanan”.

Pertimbangan konfigurasi sinkronisasi templat

Tinjau pertimbangan berikut untuk menggunakan konfigurasi sinkronisasi templat.

- Repositori harus tidak lebih besar dari 250 MB.
- Untuk mengonfigurasi sinkronisasi templat, pertama-tama tautkan repositori ke. AWS Proton Untuk informasi selengkapnya, lihat the section called “Buat tautan repositori”.
- Ketika versi template baru dibuat dari template yang disinkronkan, itu dalam `DRAFT` status.
- Versi minor baru dari template dibuat jika salah satu dari berikut ini benar:
  - Isi bundel template berbeda dari versi minor template yang disinkronkan terakhir.
  - Versi minor template terakhir yang disinkronkan sebelumnya telah dihapus.
- Sinkronisasi tidak dapat dijeda.
- Baik versi minor atau mayor baru disinkronkan secara otomatis.
- Template tingkat atas baru tidak dapat dibuat dengan konfigurasi sinkronisasi templat.
- Anda tidak dapat menyinkronkan ke satu templat dari beberapa repositori dengan konfigurasi sinkronisasi templat.
- Anda tidak dapat menggunakan tag alih-alih cabang.
- Saat Anda membuat templat layanan, Anda menentukan templat lingkungan yang kompatibel.
- Anda dapat membuat template lingkungan dan menambahkannya sebagai lingkungan yang kompatibel untuk template layanan Anda dalam komit yang sama.
- Sinkronisasi ke satu template versi utama dijalankan satu per satu. Selama sinkronisasi, jika ada komit baru yang terdeteksi, komit tersebut akan dikumpulkan dan diterapkan di akhir sinkronisasi aktif. Sinkronisasi ke berbagai versi utama template terjadi secara paralel.

- Jika Anda mengubah cabang yang disinkronkan templat Anda, sinkronisasi apa pun yang sedang berlangsung dari cabang lama terlebih dahulu selesai. Kemudian sinkronisasi dimulai dari cabang baru.
- Jika Anda mengubah repositori dari sinkronisasi templat Anda, sinkronisasi apa pun yang sedang berlangsung dari repositori lama mungkin gagal atau berjalan hingga selesai. Itu tergantung pada tahap sinkronisasi mana mereka berada.

Untuk informasi selengkapnya, lihat Referensi API AWS Proton Layanan.

Topik

- Buat konfigurasi sinkronisasi templat
- Lihat detail konfigurasi sinkronisasi templat
- Mengedit konfigurasi sinkronisasi templat
- Hapus konfigurasi sinkronisasi templat

Buat konfigurasi sinkronisasi templat

Pelajari cara membuat konfigurasi sinkronisasi templat dengan AWS Proton.

Buat prasyarat konfigurasi sinkronisasi templat:

- Anda telah menautkan repositori dengan. AWS Proton
- Sebuah bundel template terletak di repositori Anda.

Tautan repositori terdiri dari yang berikut:

- CodeConnections Koneksi yang memberikan AWS Proton izin untuk mengakses repositori Anda dan berlangganan notifikasinya.
- Peran terkait layanan. Saat Anda menautkan repositori Anda, peran terkait layanan dibuat untuk Anda.

Sebelum Anda membuat konfigurasi sinkronisasi template pertama Anda, dorong bundel template ke repositori Anda seperti yang ditunjukkan dalam tata letak direktori berikut.

```
/templates/                                          # subdirectory (optional)
```

```
/templates/my-env-template/                                  # template name
/templates/my-env-template/v1/                               # template version
/templates/my-env-template/v1/infrastructure/                # template bundle
/templates/my-env-template/v1/schema/
```

Setelah Anda membuat konfigurasi sinkronisasi templat pertama Anda, versi templat baru akan dibuat secara otomatis saat Anda mendorong komit yang menambahkan bundel templat yang diperbarui di bawah versi baru (misalnya, di bawah`/my-env-template/v2/`).

```
/templates/                                                  # subdirectory (optional)
/templates/my-env-template/                                  # template name
/templates/my-env-template/v1/                               # template version
/templates/my-env-template/v1/infrastructure/                # template bundle
/templates/my-env-template/v1/schema/
/templates/my-env-template/v2/
/templates/my-env-template/v2/infrastructure/
/templates/my-env-template/v2/schema/
```

Anda dapat menyertakan versi bundel template baru untuk satu atau beberapa templat yang dikonfigurasi sinkronisasi dalam satu komit. AWS Proton membuat versi template baru untuk setiap versi bundel template baru yang disertakan dalam komit.

Setelah membuat konfigurasi sinkronisasi templat, Anda masih dapat membuat versi template baru secara manual di konsol atau AWS CLI dengan mengunggah bundel templat dari bucket S3. Sinkronisasi template hanya berfungsi dalam satu arah: dari repositori Anda ke. AWS Proton Versi template yang dibuat secara manual tidak disinkronkan.

Setelah Anda menyiapkan konfigurasi sinkronisasi templat, AWS Proton dengarkan perubahan pada repositori Anda. Setiap kali perubahan didorong, ia mencari direktori yang memiliki nama yang sama dengan template Anda. Kemudian terlihat di dalam direktori itu untuk direktori apa pun yang terlihat seperti versi utama. AWS Proton mendaftarkan bundel template ke versi utama template yang sesuai. Versi baru selalu di `DRAFT` negara bagian. Anda dapat mempublikasikan versi baru dengan konsol atau AWS CLI.

Misalnya, Anda memiliki template yang disebut `my-env-template` dikonfigurasi untuk `my-repo/templates` disinkronkan dari cabang `main` dengan tata letak berikut.

```
/code
/code/service.go
README.md
```

```
/templates/
/templates/my-env-template/
/templates/my-env-template/v1/
/templates/my-env-template/v1/infrastructure/
/templates/my-env-template/v1/schema/
/templates/my-env-template/v2/
/templates/my-env-template/v2/infrastructure/
/templates/my-env-template/v2/schema/
```

AWS Proton menyinkronkan isi dari `/templates/my-env-template/v1/` to `my-env-template:1` dan isi dari `/templates/my-env-template/v2/` tomy`-env-template:2`. Jika mereka belum ada, itu menciptakan versi utama ini.

AWS Proton menemukan direktori pertama yang cocok dengan nama template. Anda dapat membatasi AWS Proton pencarian direktori dengan menentukan `subdirectoryPath` kapan Anda membuat atau mengedit konfigurasi sinkronisasi template. Misalnya, Anda dapat menentukan `/production-templates/` untuk`subdirectoryPath`.

Anda dapat membuat konfigurasi sinkronisasi template menggunakan konsol atau CLI.

Konsol Manajemen AWS

Buat konfigurasi sinkronisasi templat dan templat menggunakan konsol.

1. Di AWS Proton konsol, pilih Template lingkungan.
2. Pilih Buat template lingkungan.
3. Di halaman Buat templat lingkungan, di bagian Opsi templat, pilih Buat templat untuk menyediakan lingkungan baru.
4. Di bagian Sumber bundel Template, pilih Sinkronkan templat dari Git.
5. Di bagian repositori kode sumber:
    a. Untuk Repositori, pilih repositori tertaut yang berisi bundel template Anda.
    b. Untuk Branch, pilih cabang repositori untuk disinkronkan.
    c. (Opsional) Untuk direktori bundel Template, masukkan nama direktori untuk mencakup pencarian bundel template Anda.
6. Di bagian Detail Template.
    a. Masukkan nama Template.
    b. (Opsional) Masukkan nama tampilan Template.

c. (Opsional) Masukkan deskripsi Template untuk template lingkungan.

7. (Opsional) Centang kotak centang untuk Sesuaikan pengaturan enkripsi (lanjutan) di bagian Pengaturan enkripsi untuk menyediakan kunci enkripsi Anda sendiri.
8. (Opsional) Di bagian Tag, pilih Tambahkan tag baru dan masukkan kunci dan nilai untuk membuat tag terkelola pelanggan.
9. Pilih template Create Environment.

   Anda sekarang berada di halaman baru yang menampilkan status dan detail untuk template lingkungan baru Anda. Rincian ini mencakup daftar tag yang AWS dikelola dan dikelola pelanggan. AWS Proton secara otomatis menghasilkan tag AWS terkelola untuk Anda saat Anda membuat AWS Proton sumber daya. Untuk informasi selengkapnya, lihat AWS Proton sumber daya dan penandaan.
10. Di halaman detail templat, pilih tab Sinkronisasi untuk melihat data detail konfigurasi sinkronisasi templat.
11. Pilih tab Versi templat untuk melihat versi templat dengan detail status.
12. Status status template lingkungan baru dimulai di status Draft. Anda dan orang lain dengan `proton:CreateEnvironment` izin dapat melihat dan mengaksesnya. Ikuti langkah selanjutnya untuk membuat template tersedia untuk orang lain.
13. Di bagian Versi Template, pilih tombol radio di sebelah kiri versi minor template yang baru saja Anda buat (1.0). Sebagai alternatif, Anda dapat memilih Publikasikan di peringatan info dan lewati langkah berikutnya.
14. Di bagian Versi templat, pilih Publikasikan.
15. Status template berubah menjadi Diterbitkan. Ini adalah versi terbaru dan Direkomendasikan dari template.
16. Di panel navigasi, pilih Template lingkungan untuk melihat daftar templat dan detail lingkungan Anda.

Prosedur untuk membuat templat layanan dan konfigurasi sinkronisasi templat serupa.

AWS CLI

Buat konfigurasi sinkronisasi templat dan templat menggunakan AWS CLI.

1. Buat template. Dalam contoh ini, template lingkungan dibuat.

   Jalankan perintah berikut.

```
$ aws proton create-environment-template \
    --name "env-template"
```

Responsnya adalah sebagai berikut.

```
{
    "environmentTemplate": {
        "arn": "arn:aws:proton:us-east-1:123456789012:environment-template/env-template",
        "createdAt": "2021-11-07T23:32:43.045000+00:00",
        "displayName": "env-template",
        "lastModifiedAt": "2021-11-07T23:32:43.045000+00:00",
        "name": "env-template",
        "status": "DRAFT",
        "templateName": "env-template"
    }
}
```

2. Buat konfigurasi sinkronisasi template Anda AWS CLI dengan memberikan yang berikut:

   - Template yang ingin Anda sinkronkan. Setelah Anda membuat konfigurasi sinkronisasi templat, Anda masih dapat membuat versi baru darinya secara manual di konsol atau dengan AWS CLI.
   - Nama template.
   - Jenis template.
   - Repositori tertaut yang ingin Anda sinkronkan.
   - Penyedia repositori tertaut.
   - Cabang tempat bundel templat berada.
   - (Opsional) Jalur ke direktori yang berisi bundel template Anda. Secara default, AWS Proton cari direktori pertama yang cocok dengan nama template Anda.

   Jalankan perintah berikut.

```
$ aws proton create-template-sync-config \
    --template-name "env-template" \
    --template-type "ENVIRONMENT" \
    --repository-name "myrepos/templates" \
    --repository-provider "GITHUB" \
```

```
    --branch "main" \
    --subdirectory "env-template/"
```

Responsnya adalah sebagai berikut.

```
{
    "templateSyncConfigDetails": {
        "branch": "main",
        "repositoryName": "myrepos/templates",
        "repositoryProvider": "GITHUB",
        "subdirectory": "templates",
        "templateName": "env-template",
        "templateType": "ENVIRONMENT"
    }
}
```

3. Untuk mempublikasikan versi template Anda, lihatDaftarkan dan terbitkan templat.

Menyinkronkan templat layanan

Contoh sebelumnya menunjukkan bagaimana Anda dapat menyinkronkan templat lingkungan. Template layanan serupa. Untuk menyinkronkan templat layanan, Anda menambahkan file tambahan bernama `.template-registration.yaml` ke setiap direktori versi utama dalam bundel templat Anda. File ini berisi detail tambahan yang AWS Proton diperlukan saat membuat versi template layanan untuk Anda mengikuti komit. Saat Anda secara eksplisit membuat versi template layanan menggunakan AWS Proton konsol atau API, Anda memberikan detail ini sebagai input, dan file ini menggantikan input ini untuk sinkronisasi templat.

```
./templates/                                              # subdirectory (optional)
 /templates/my-svc-template/                              # service template name
 /templates/my-svc-template/v1/                           # service template version
 /templates/my-svc-template/v1/.template-registration.yaml   # service template version
 properties
 /templates/my-svc-template/v1/instance_infrastructure/      # template bundle
 /templates/my-svc-template/v1/schema/
```

`.template-registration.yaml`File berisi rincian berikut:

- Lingkungan yang kompatibel [wajib] - Lingkungan berdasarkan templat lingkungan ini dan versi utama kompatibel dengan layanan berdasarkan versi templat layanan ini.

- Sumber komponen yang didukung [opsional] - Komponen yang menggunakan sumber ini kompatibel dengan layanan berdasarkan versi templat layanan ini. Jika tidak ditentukan, komponen tidak dapat dilampirkan ke layanan ini. Untuk informasi selengkapnya tentang komponen, lihatKomponen-komponen.

Sintaks YAMB file adalah sebagai berikut:

```
compatible_environments:
  - env-templ-name:major-version
  - ...
supported_component_sources:
  - DIRECTLY_DEFINED
```

Tentukan satu atau lebih template lingkungan/kombinasi versi utama. Menentukan `supported_component_sources` adalah opsional, dan satu-satunya nilai yang didukung adalah`DIRECTLY_DEFINED`.

Example.template-registration.yaml

Dalam contoh ini, versi template layanan kompatibel dengan versi utama 1 dan 2 dari template `my-env-template` lingkungan. Ini juga kompatibel dengan versi utama 1 dan 3 dari template `another-env-template` lingkungan. File tidak ditentukan`supported_component_sources`, sehingga komponen tidak dapat dilampirkan ke layanan berdasarkan versi templat layanan ini.

```
compatible_environments:
  - my-env-template:1
  - my-env-template:2
  - another-env-template:1
  - another-env-template:3
```

> **Note**
>
> Sebelumnya, AWS Proton didefinisikan file yang berbeda,`.compatible-envs`, untuk menentukan lingkungan yang kompatibel. AWS Proton masih mendukung file itu dan formatnya untuk kompatibilitas mundur. Kami tidak menyarankan menggunakannya lagi, karena tidak dapat diperluas dan tidak dapat mendukung fitur yang lebih baru seperti komponen.

Lihat detail konfigurasi sinkronisasi templat

Lihat data detail konfigurasi sinkronisasi templat menggunakan konsol atau CLI.

Konsol Manajemen AWS

Gunakan konsol untuk melihat detail konfigurasi sinkronisasi templat.

1. Di panel navigasi, pilih templat (Lingkungan atau Layanan).
2. Untuk melihat data detail, pilih nama templat tempat Anda membuat konfigurasi sinkronisasi templat.
3. Di halaman detail untuk templat, pilih tab Sinkronisasi untuk melihat data detail konfigurasi sinkronisasi templat.

AWS CLI

Gunakan AWS CLI untuk melihat template yang disinkronkan.

Jalankan perintah berikut.

```
$ aws proton get-template-sync-config \
    --template-name "svc-template" \
    --template-type "SERVICE"
```

Responsnya adalah sebagai berikut.

```
{
    "templateSyncConfigDetails": {
        "branch": "main",
        "repositoryProvider": "GITHUB",
        "repositoryName": "myrepos/myrepo",
        "subdirectory": "svc-template",
        "templateName": "svc-template",
        "templateType": "SERVICE"
    }
}
```

Gunakan AWS CLI untuk mendapatkan status sinkronisasi templat.

Untuk`template-version`, masukkan template versi mayor.

Jalankan perintah berikut.

```
$ aws proton get-template-sync-status \
    --template-name "env-template" \
    --template-type "ENVIRONMENT" \
    --template-version "1"
```

Mengedit konfigurasi sinkronisasi templat

Anda dapat mengedit salah satu parameter konfigurasi sinkronisasi templat kecuali `template-name` dan`template-type`.

Pelajari cara mengedit konfigurasi sinkronisasi templat menggunakan konsol atau CLI.

Konsol Manajemen AWS

Edit cabang konfigurasi sinkronisasi templat menggunakan konsol.

Dalam daftar template.

1. Di AWS Proton konsol, pilih Template (Lingkungan atau Layanan).
2. Dalam daftar templat, pilih nama templat dengan konfigurasi sinkronisasi templat yang ingin Anda edit.
3. Di halaman detail templat, pilih tab Sinkronisasi templat.
4. Di bagian Detail sinkronisasi templat, pilih Edit.
5. Di halaman Edit, di bagian repositori kode sumber, untuk Cabang, pilih cabang, lalu pilih Simpan konfigurasi.

AWS CLI

Contoh perintah dan respons berikut menunjukkan bagaimana Anda dapat mengedit konfigurasi sinkronisasi template **`branch`** menggunakan CLI.

Jalankan perintah berikut.

```
$ aws proton update-template-sync-config \
    --template-name "env-template" \
    --template-type "ENVIRONMENT" \
    --repository-provider "GITHUB" \
```

```
    --repository-name "myrepos/templates" \
    --branch "fargate" \
    --subdirectory "env-template"
```

Responsnya adalah sebagai berikut.

```
{
    "templateSyncConfigDetails": {
        "branch": "fargate",
        "repositoryProvider": "GITHUB",
        "repositoryName": "myrepos/myrepo",
        "subdirectory": "templates",
        "templateName": "env-template",
        "templateType": "ENVIRONMENT"
    }
}
```

Anda juga dapat menggunakan AWS CLI untuk memperbarui template layanan yang disinkronkan.

Hapus konfigurasi sinkronisasi templat

Hapus konfigurasi sinkronisasi templat menggunakan konsol atau CLI.

Konsol Manajemen AWS

Hapus konfigurasi sinkronisasi templat menggunakan konsol.

1. Di halaman detail templat, pilih tab Sinkronisasi.
2. Di bagian Sinkronkan detail, pilih Putuskan sambungan.

AWS CLI

Contoh perintah dan tanggapan berikut menunjukkan cara menggunakan konfigurasi template yang disinkronkan AWS CLI untuk menghapus.

Jalankan perintah berikut.

```
$ aws proton delete-template-sync-config \
    --template-name "env-template" \
```

```
    --template-type "ENVIRONMENT"
```

Responsnya adalah sebagai berikut.

```
{
    "templateSyncConfig": {
        "templateName": "env-template",
        "templateType": "ENVIRONMENT"
    }
}
```

Konfigurasi sinkronisasi layanan

Dengan sinkronisasi layanan, Anda dapat mengonfigurasi dan menerapkan AWS Proton layanan Anda menggunakan Git. Anda dapat menggunakan sinkronisasi layanan untuk mengelola penerapan awal dan pembaruan ke AWS Proton layanan Anda dengan konfigurasi yang ditentukan dalam repositori Git. Melalui Git, Anda dapat menggunakan fitur seperti pelacakan versi dan permintaan tarik untuk mengonfigurasi, mengelola, dan menyebarkan layanan Anda. Service sync menggabungkan AWS Proton dan Git untuk membantu Anda menyediakan infrastruktur standar yang didefinisikan dan dikelola melalui AWS Proton template. Ini mengelola definisi layanan di repositori Git Anda dan mengurangi peralihan alat. Dibandingkan dengan menggunakan Git saja, standarisasi template dan penerapan di AWS Proton membantu Anda menghabiskan lebih sedikit waktu mengelola infrastruktur Anda. AWS Proton juga memberikan transparansi dan auditabilitas yang lebih tinggi untuk pengembang dan tim platform.

AWS Proton Berkas OPS

`proton-ops`File menentukan tempat AWS Proton menemukan file spesifikasi yang digunakan untuk memperbarui instance layanan Anda. Ini juga mendefinisikan urutan apa untuk memperbarui instance layanan dan kapan harus mempromosikan perubahan dari satu instance ke instance lainnya.

`proton-ops`File ini mendukung sinkronisasi instance layanan menggunakan file spesifikasi, atau beberapa file spesifikasi, yang ditemukan di repositori tertaut Anda. Anda dapat melakukan ini dengan mendefinisikan blok sinkronisasi dalam `proton-ops` file, seperti pada contoh berikut.

Contoh. /konfigurasi/proton-ops.yaml:

```
sync:
```

```
  services:
      frontend-svc:
          alpha:
              branch: dev
              spec: ./frontend-svc/test/frontend-spec.yaml
          beta:
              branch: dev
              spec: ./frontend-svc/test/frontend-spec.yaml
          gamma:
              branch: pre-prod
              spec: ./frontend-svc/pre-prod/frontend-spec.yaml
          prod-one:
              branch: prod
              spec: ./frontend-svc/prod/frontend-spec-second.yaml
          prod-two:
              branch: prod
              spec: ./frontend-svc/prod/frontend-spec-second.yaml
          prod-three:
              branch: prod
              spec: ./frontend-svc/prod/frontend-spec-second.yaml
```

Dalam contoh sebelumnya, `frontend-svc` adalah nama layanan, dan,,`alpha`,`beta`, `gamma` `prod-one``prod-two`, dan `prod-three` merupakan contoh.

`spec`File dapat berupa semua instance atau subset dari instance yang didefinisikan dalam file. `proton-ops` Namun, setidaknya, itu harus memiliki instance yang ditentukan di dalam cabang dan spesifikasi yang disinkronkannya. Jika instance tidak ditentukan dalam `proton-ops` file, dengan cabang dan lokasi `spec` file tertentu, sinkronisasi layanan tidak akan membuat atau memperbarui instance tersebut.

Contoh berikut menunjukkan seperti apa `spec` file tersebut. Ingat, `proton-ops` file tersebut disinkronkan dari `spec` file-file ini.

Contoh**`./frontend-svc/test/frontend-spec.yaml`**:

```
proton: "ServiceSpec"
instances:
- name: "alpha"
  environment: "frontend-env"
  spec:
    port: 80
    desired_count: 1
```

```
    task_size: "x-small"
    image: "public.ecr.aws/z9d2n7e1/nginx:1.21.0"
- name: "beta"
  environment: "frontend-env"
  spec:
    port: 80
    desired_count: 1
    task_size: "x-small"
    image: "public.ecr.aws/z9d2n7e1/nginx:1.21.0"
```

Contoh **`./frontend-svc/pre-prod/frontend-spec.yaml`**:

```
proton: "ServiceSpec"
instances:
- name: "gamma"
  environment: "frontend-env"
  spec:
    port: 80
    desired_count: 1
    task_size: "x-small"
    image: "public.ecr.aws/z9d2n7e1/nginx:1.21.0"
```

Contoh **`./frontend-svc/prod/frontend-spec-second.yaml`**:

```
proton: "ServiceSpec"
instances:
- name: "prod-one"
  environment: "frontend-env"
  spec:
    port: 80
    desired_count: 1
    task_size: "x-small"
    image: "public.ecr.aws/z9d2n7e1/nginx:1.21.0"
- name: "prod-two"
  environment: "frontend-env"
  spec:
    port: 80
    desired_count: 1
    task_size: "x-small"
    image: "public.ecr.aws/z9d2n7e1/nginx:1.21.0"
- name: "prod-three"
  environment: "frontend-env"
  spec:
```

```
    port: 80
    desired_count: 1
    task_size: "x-small"
    image: "public.ecr.aws/z9d2n7e1/nginx:1.21.0"
```

Jika sebuah instance tidak disinkronkan, dan ada masalah berkelanjutan saat mencoba menyinkronkannya, memanggil `GetServiceInstanceSyncStatus`API dapat membantu menyelesaikan masalah.

> **Note**
>
> Pelanggan yang menggunakan sinkronisasi layanan masih dibatasi oleh AWS Proton batasan.

Pemblokir

Dengan menyinkronkan layanan Anda menggunakan sinkronisasi AWS Proton layanan, Anda dapat memperbarui spesifikasi layanan Anda dan membuat serta memperbarui instance layanan dari repositori Git Anda. Namun, mungkin ada saat-saat di mana Anda perlu memperbarui layanan atau instance secara manual melalui Konsol Manajemen AWS atau AWS CLI.

AWS Proton membantu menghindari menimpa setiap perubahan manual yang Anda buat melalui Konsol Manajemen AWS atau AWS CLI, seperti memperbarui instance layanan atau menghapus instance layanan. Untuk mencapai hal ini, AWS Proton secara otomatis membuat pemblokir sinkronisasi layanan dengan menonaktifkan sinkronisasi layanan saat mendeteksi perubahan manual.

Untuk mendapatkan semua pemblokir yang terkait dengan layanan, Anda harus melakukan hal berikut agar masing-masing `serviceInstance` terkait dengan layanan:

- Panggil `getServiceSyncBlockerSummary` API hanya dengan file`serviceName`.
- Panggil `getServiceSyncBlockerSummary` API dengan `serviceName` dan`serviceInstanceName`.

Ini mengembalikan daftar pemblokir terbaru dan status yang terkait dengannya. Jika ada pemblokir yang ditandai AKTIF, Anda harus menyelesaikannya dengan memanggil `UpdateServiceSyncBlocker` API dengan `blockerId` dan `resolvedReason` untuk masing-masing pemblokir.

Jika Anda memperbarui atau membuat instance layanan secara manual, AWS Proton buat pemblokir sinkronisasi layanan pada instance layanan. AWS Proton terus menyinkronkan semua instance layanan lainnya, tetapi menonaktifkan sinkronisasi instance layanan ini hingga pemblokir diselesaikan. Jika Anda menghapus instance layanan dari layanan, AWS Proton buat pemblokir sinkronisasi layanan pada layanan. Ini AWS Proton mencegah sinkronisasi salah satu instance layanan hingga pemblokir telah diselesaikan.

Setelah Anda memiliki semua pemblokir aktif, Anda harus menyelesaikannya dengan memanggil `UpdateServiceSyncBlocker` API dengan `blockerId` dan `resolvedReason` untuk masing-masing pemblokir aktif.

Dengan menggunakan Konsol Manajemen AWS, Anda dapat menentukan apakah sinkronisasi layanan dinonaktifkan dengan menavigasi ke AWS Proton dan memilih tab Sinkronisasi Layanan. Jika instance layanan atau layanan diblokir, tombol Aktifkan akan muncul. Untuk mengaktifkan sinkronisasi layanan, pilih Aktifkan.

Topik

- Buat konfigurasi sinkronisasi layanan
- Melihat detail konfigurasi untuk sinkronisasi layanan
- Mengedit konfigurasi sinkronisasi layanan
- Menghapus konfigurasi sinkronisasi layanan

Buat konfigurasi sinkronisasi layanan

Anda dapat membuat konfigurasi sinkronisasi layanan menggunakan konsol atau AWS CLI.

Konsol Manajemen AWS

1. Pada halaman Choose a service template, pilih template dan pilih Configure.
2. Pada halaman Konfigurasi layanan, di bagian Detail layanan, masukkan nama Layanan baru.
3. (Opsional) Masukkan deskripsi untuk layanan.
4. Di bagian Repositori kode sumber Aplikasi, pilih Pilih repositori Git tertaut untuk memilih repositori yang telah Anda tautkan. AWS Proton Jika Anda belum memiliki repositori tertaut, pilih Tautkan repositori Git lain dan ikuti petunjuk di Buat tautan ke repositori Anda.
5. Untuk Repositori, pilih nama repositori kode sumber Anda dari daftar.
6. Untuk Branch, pilih nama cabang repositori untuk kode sumber Anda dari daftar.

7. (Opsional) Di bagian Tag, pilih Tambahkan tag baru dan masukkan kunci dan nilai untuk membuat tag yang dikelola pelanggan.
8. Pilih Berikutnya.
9. Pada halaman Konfigurasi instance layanan, di bagian Sumber definisi layanan, pilih Sinkronkan layanan Anda dari Git.
10. Di bagian File definisi layanan, jika Anda AWS Proton ingin membuat `proton-ops` file Anda, pilih Saya ingin AWS Proton membuat file. Dengan opsi ini, AWS Proton buat `proton-ops` file `spec` dan di lokasi yang Anda tentukan. Pilih Saya menyediakan file saya sendiri untuk membuat file OPS Anda sendiri.
11. Di bagian repositori definisi Layanan, pilih Pilih repositori Git tertaut untuk memilih repositori yang telah Anda tautkan. AWS Proton
12. Untuk nama Repositori, pilih nama repositori kode sumber Anda dari daftar.
13. Untuk cabang **`proton-ops`** file, pilih nama cabang Anda dari daftar di mana AWS Proton akan menempatkan OPS dan file spesifikasi Anda.
14. Di bagian Contoh layanan, setiap bidang diisi secara otomatis berdasarkan nilai dalam `proton-ops` file.
15. Pilih Berikutnya dan tinjau masukan Anda.
16. Pilih Buat.

AWS CLI

Buat konfigurasi sinkronisasi layanan menggunakan AWS CLI

- Jalankan perintah berikut.

```
$ aws proton create-service-sync-config \
    --resource "service-arn" \
    --repository-provider "GITHUB" \
    --repository "example/proton-sync-service" \
    --ops-file-branch "main" \
    --proton-ops-file "./configuration/custom-proton-ops.yaml" (optional)
```

Responsnya adalah sebagai berikut.

```
{
    "serviceSyncConfig": {
```

```
        "branch": "main",
        "filePath": "./configuration/custom-proton-ops.yaml",
        "repositoryName": "example/proton-sync-service",
        "repositoryProvider": "GITHUB",
        "serviceName": "service name"
    }
}
```

Melihat detail konfigurasi untuk sinkronisasi layanan

Anda dapat melihat data detail konfigurasi untuk sinkronisasi layanan menggunakan konsol atau AWS CLI.

Konsol Manajemen AWS

Menggunakan konsol untuk melihat detail konfigurasi untuk sinkronisasi layanan

1. Pada panel navigasi, silakan pilih Layanan.
2. Untuk melihat data detail, pilih nama layanan yang Anda buat untuk konfigurasi sinkronisasi layanan.
3. Di halaman detail untuk layanan, pilih tab Sinkronisasi layanan untuk melihat data detail konfigurasi untuk sinkronisasi layanan.

AWS CLI

Gunakan AWS CLI untuk mendapatkan layanan yang disinkronkan.

Jalankan perintah berikut.

```
$ aws proton get-service-sync-config \
    --service-name "service name"
```

Responsnya adalah sebagai berikut.

```
{
    "serviceSyncConfig": {
        "branch": "main",
        "filePath": "./configuration/custom-proton-ops.yaml",
        "repositoryName": "example/proton-sync-service",
```

```
        "repositoryProvider": "GITHUB",
        "serviceName": "service name"
    }
}
```

Gunakan AWS CLI untuk mendapatkan status sinkronisasi layanan.

Jalankan perintah berikut.

```
$ aws proton get-service-sync-status \
    --service-name "service name"
```

Mengedit konfigurasi sinkronisasi layanan

Anda dapat mengedit konfigurasi sinkronisasi layanan menggunakan konsol atau AWS CLI.

Konsol Manajemen AWS

Edit konfigurasi sinkronisasi layanan menggunakan konsol.

1. Pada panel navigasi, silakan pilih Layanan.
2. Untuk melihat data detail, pilih nama layanan yang Anda buat untuk konfigurasi sinkronisasi layanan.
3. Pada halaman detail layanan, pilih tab Sinkronisasi layanan.
4. Di bagian Sinkronisasi layanan, pilih Edit.
5. Pada halaman Edit, perbarui informasi yang ingin Anda edit, lalu pilih Simpan.

AWS CLI

Contoh perintah dan respons berikut menunjukkan bagaimana Anda dapat mengedit konfigurasi sinkronisasi layanan menggunakan AWS CLI.

Jalankan perintah berikut.

```
$ aws proton update-service-sync-config \
    --service-name "service name" \
    --repository-provider "GITHUB" \
    --repository "example/proton-sync-service" \
```

```
    --ops-file-branch "main" \
    --ops-file "./configuration/custom-proton-ops.yaml"
```

Responsnya adalah sebagai berikut.

```
{
    "serviceSyncConfig": {
        "branch": "main",
        "filePath": "./configuration/custom-proton-ops.yaml",
        "repositoryName": "example/proton-sync-service",
        "repositoryProvider": "GITHUB",
        "serviceName": "service name"
    }
}
```

Menghapus konfigurasi sinkronisasi layanan

Anda dapat menghapus konfigurasi sinkronisasi layanan menggunakan konsol atau AWS CLI.

Konsol Manajemen AWS

Menghapus konfigurasi sinkronisasi layanan menggunakan konsol

1. Pada halaman detail layanan, pilih tab Sinkronisasi layanan.
2. Di bagian Detail sinkronisasi layanan, pilih Putuskan sambungan untuk memutuskan sambungan repositori Anda. Setelah repositori Anda terputus, kami tidak lagi menyinkronkan layanan dari repositori itu.

AWS CLI

Contoh perintah dan tanggapan berikut menunjukkan cara menggunakan konfigurasi yang disinkronkan AWS CLI untuk menghapus layanan.

Jalankan perintah berikut.

```
$ aws proton delete-service-sync-config \
    --service-name "service name"
```

Responsnya adalah sebagai berikut.

```
{
    "serviceSyncConfig": {
        "branch": "main",
        "filePath": "./configuration/custom-proton-ops.yaml",
        "repositoryName": "example/proton-sync-service",
        "repositoryProvider": "GITHUB",
        "serviceName": "service name"
    }
}
```

> **Note**
>
> Sinkronisasi layanan tidak menghapus instance layanan. Itu hanya menghapus konfigurasi.

AWS Proton lingkungan

Untuk AWS Proton, lingkungan mewakili kumpulan sumber daya dan kebijakan bersama yang digunakan AWS Proton layanan. Mereka dapat berisi sumber daya apa pun yang diharapkan untuk dibagikan di seluruh instance AWS Proton layanan. Sumber daya ini dapat mencakup VPCs, cluster, dan penyeimbang beban bersama atau API Gateways. AWS Proton Lingkungan harus dibuat sebelum layanan dapat digunakan untuk itu.

Bagian ini menjelaskan cara mengelola lingkungan menggunakan operasi membuat, melihat, memperbarui, dan menghapus. Untuk >informasi tambahan, lihat Referensi API AWS Proton Layanan.

Topik

- Peran IAM
- Buat lingkungan
- Lihat data lingkungan
- Perbarui lingkungan
- Hapus lingkungan
- Koneksi akun lingkungan
- Lingkungan yang dikelola pelanggan
- CodeBuild pembuatan peran penyediaan

Peran IAM

Dengan AWS Proton, Anda menyediakan peran dan AWS KMS kunci IAM untuk AWS sumber daya yang Anda miliki dan kelola. Ini kemudian diterapkan dan digunakan oleh sumber daya yang dimiliki dan dikelola oleh pengembang. Anda membuat peran IAM untuk mengontrol akses tim pengembang Anda ke AWS Proton API.

AWS Proton peran layanan

Saat Anda membuat lingkungan baru, Anda menyediakan peran layanan IAM terkait. Peran berisi semua izin yang diperlukan untuk memperbarui semua infrastruktur yang disediakan yang ditentukan dalam templat lingkungan dan templat layanan. Untuk contoh peran, lihatAWS Proton peran layanan untuk penyediaan menggunakan CloudFormation. Jika Anda menggunakan koneksi akun lingkungan

dan akun lingkungan, Anda membuat peran di akun lingkungan yang dipilih. Untuk informasi selengkapnya, lihat Buat lingkungan di satu akun dan ketentuan di akun lain dan Koneksi akun lingkungan.

Bagaimana Anda menyediakan peran layanan ini, dan siapa yang mengambil peran tersebut, bergantung pada metode penyediaan lingkungan Anda.

- AWS-managed provisioning — Anda memberikan peran AWS Proton, baik secara langsung saat membuat lingkungan, atau secara tidak langsung melalui koneksi akun. AWS Proton mengasumsikan peran dalam akun yang relevan dengan lingkungan penyediaan dan infrastruktur layanan.
- Penyediaan yang dikelola sendiri — Anda bertanggung jawab untuk mengonfigurasi otomatisasi penyediaan Anda untuk mengambil peran yang sesuai menggunakan kredensi yang sesuai saat permintaan tarik (PR) memicu tindakan penyediaan. Untuk contoh GitHub Tindakan yang mengasumsikan peran, lihat Mengasumsikan Peran dalam dokumentasi Tindakan Untuk Tindakan "Konfigurasi AWS Kredensial". GitHub

Untuk informasi selengkapnya tentang metode penyediaan, lihat. the section called "Metode penyediaan"

Buat lingkungan

Belajarlah untuk menciptakan AWS Proton lingkungan.

Anda dapat membuat AWS Proton lingkungan dengan salah satu dari dua cara:

- Membuat, mengelola, dan menyediakan lingkungan standar dengan menggunakan template lingkungan standar. AWS Proton menyediakan infrastruktur untuk lingkungan Anda.
- Connect AWS Proton ke infrastruktur yang dikelola pelanggan dengan menggunakan template lingkungan yang dikelola pelanggan. Anda menyediakan sumber daya bersama Anda sendiri di luar AWS Proton, dan kemudian Anda memberikan output penyediaan yang dapat AWS Proton digunakan.

Anda dapat memilih salah satu dari beberapa pendekatan penyediaan saat Anda membuat lingkungan.

- AWS penyediaan terkelola - Membuat, mengelola, dan menyediakan lingkungan dalam satu akun. AWS Proton ketentuan lingkungan Anda.

Metode ini hanya mendukung template kode CloudFormation infrastruktur (IAc).

- AWS penyediaan terkelola ke akun lain — Dalam satu akun manajemen, buat dan kelola lingkungan yang disediakan di akun lain dengan koneksi akun lingkungan. AWS Proton menyediakan lingkungan Anda di akun lain. Untuk informasi selengkapnya, lihat Buat lingkungan di satu akun dan ketentuan di akun lain dan Koneksi akun lingkungan.

  Metode ini hanya mendukung template CloudFormation IAC.

- Penyediaan yang dikelola sendiri — AWS Proton mengirimkan permintaan tarik penyediaan ke repositori tertaut dengan infrastruktur penyediaan Anda sendiri.

  Metode ini hanya mendukung template Terraform IAc.

- CodeBuild provisioning — AWS Proton digunakan AWS CodeBuild untuk menjalankan perintah shell yang Anda berikan. Perintah Anda dapat membaca input yang AWS Proton menyediakan, dan bertanggung jawab untuk penyediaan atau deprovisioning infrastruktur dan menghasilkan nilai output. Bundel template untuk metode ini menyertakan perintah Anda dalam file manifes dan program, skrip, atau file lain yang mungkin diperlukan oleh perintah ini.

  Sebagai contoh untuk menggunakan CodeBuild penyediaan, Anda dapat menyertakan kode yang menggunakan AWS sumber daya AWS Cloud Development Kit (AWS CDK) untuk menyediakan, dan manifes yang menginstal CDK dan menjalankan kode CDK Anda.

  Untuk informasi selengkapnya, lihat the section called “CodeBuild bundel”.

  > **Note**
  >
  > Anda dapat menggunakan CodeBuild penyediaan dengan lingkungan dan layanan. Saat ini Anda tidak dapat menyediakan komponen dengan cara ini.

Dengan penyediaan AWS terkelola (baik di akun yang sama maupun ke akun lain), lakukan panggilan AWS Proton langsung untuk menyediakan sumber daya Anda.

Dengan penyediaan yang dikelola sendiri, AWS Proton buat permintaan tarik untuk menyediakan file IAc terkompilasi yang digunakan mesin IAc Anda untuk menyediakan sumber daya.

Lihat informasi selengkapnya di the section called “Metode penyediaan”, the section called “Bundel template”, dan the section called “Persyaratan skema lingkungan”.

Topik

- Membuat dan menyediakan lingkungan standar di akun yang sama
- Buat lingkungan di satu akun dan ketentuan di akun lain
- Membuat dan menyediakan lingkungan menggunakan penyediaan yang dikelola sendiri

Membuat dan menyediakan lingkungan standar di akun yang sama

Gunakan konsol atau AWS CLI untuk membuat dan menyediakan lingkungan dalam satu akun. Penyediaan dikelola oleh. AWS

Konsol Manajemen AWS

Gunakan konsol untuk membuat dan menyediakan lingkungan dalam satu akun

1. Di AWS Proton konsol, pilih Lingkungan.
2. Pilih Buat lingkungan.
3. Di halaman Pilih templat lingkungan, pilih templat dan pilih Konfigurasi.
4. Di halaman Konfigurasi lingkungan, di bagian Penyediaan, pilih penyediaan AWS terkelola.
5. Di bagian Akun Deployment, pilih Ini Akun AWS.
6. Di halaman Konfigurasi lingkungan, di bagian Pengaturan lingkungan, masukkan nama Lingkungan.
7. (Opsional) Masukkan deskripsi untuk lingkungan.
8. Di bagian Peran lingkungan, pilih peran AWS Proton layanan yang Anda buat sebagai bagian darinyaMenyiapkan peran AWS Proton layanan.
9. (Opsional) Di bagian peran Komponen, pilih peran layanan yang memungkinkan komponen yang didefinisikan secara langsung untuk berjalan di lingkungan dan cakupan sumber daya yang dapat mereka sediakan. Untuk informasi selengkapnya, lihat Komponen-komponen.
10. (Opsional) Di bagian Tag, pilih Tambahkan tag baru dan masukkan kunci dan nilai untuk membuat tag terkelola pelanggan.
11. Pilih Berikutnya.
12. Di halaman Konfigurasi pengaturan khusus lingkungan, Anda harus memasukkan nilai untuk `required` parameter. Anda dapat memasukkan nilai untuk `optional` parameter atau menggunakan default saat diberikan.

13. Pilih Berikutnya dan tinjau masukan Anda.
14. Pilih Buat.

    Lihat detail dan status lingkungan, serta tag AWS terkelola dan tag terkelola pelanggan untuk lingkungan Anda.
15. Pada panel navigasi, pilih Lingkungan.

    Halaman baru menampilkan daftar lingkungan Anda bersama dengan status dan detail lingkungan lainnya.

AWS CLI

Gunakan AWS CLI untuk membuat dan menyediakan lingkungan dalam satu akun.

Untuk membuat lingkungan, Anda menentukan ARN peran AWS Proton layanan, jalur ke file spesifikasi Anda, nama lingkungan, ARN template lingkungan, versi mayor dan minor, dan deskripsi (opsional).

Contoh berikutnya menunjukkan file spesifikasi YAML diformat yang menentukan nilai untuk dua input yang didefinisikan dalam file skema template lingkungan. Anda dapat menggunakan `get-environment-template-minor-version` perintah untuk melihat skema template lingkungan.

```
proton: EnvironmentSpec
spec:
  my_sample_input: "the first"
  my_other_sample_input: "the second"
```

Buat lingkungan dengan menjalankan perintah berikut.

```
$ aws proton create-environment \
    --name "MySimpleEnv" \
    --template-name simple-env \
    --template-major-version 1 \
    --proton-service-role-arn "arn:aws:iam::123456789012:role/AWS ProtonServiceRole"
 \
    --spec "file://env-spec.yaml"
```

Respons:

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/MySimpleEnv",
        "createdAt": "2020-11-11T23:03:05.405000+00:00",
        "deploymentStatus": "IN_PROGRESS",
        "lastDeploymentAttemptedAt": "2020-11-11T23:03:05.405000+00:00",
        "name": "MySimpleEnv",
        "protonServiceRoleArn": "arn:aws:iam::123456789012:role/ProtonServiceRole",
        "templateName": "simple-env"
    }
}
```

Setelah Anda membuat lingkungan baru, Anda dapat melihat daftar AWS dan tag terkelola pelanggan seperti yang ditunjukkan dalam perintah contoh berikut. AWS Proton secara otomatis menghasilkan tag AWS terkelola untuk Anda. Anda juga dapat memodifikasi dan membuat tag yang dikelola pelanggan menggunakan AWS CLI. Untuk informasi selengkapnya, lihat AWS Proton sumber daya dan penandaan.

Perintah:

```
$ aws proton list-tags-for-resource \
    --resource-arn "arn:aws:proton:region-id:123456789012:environment/MySimpleEnv"
```

Buat lingkungan di satu akun dan ketentuan di akun lain

Gunakan konsol atau AWS CLI untuk membuat lingkungan standar di akun manajemen yang menyediakan infrastruktur lingkungan di akun lain. Penyediaan dikelola oleh. AWS

Sebelum menggunakan konsol atau CLI, selesaikan langkah-langkah berikut.

1. Identifikasi akun Akun AWS IDs untuk manajemen dan lingkungan, dan salin untuk digunakan nanti.
2. Di akun lingkungan, buat peran AWS Proton layanan dengan izin minimum untuk lingkungan yang akan dibuat. Untuk informasi selengkapnya, lihat AWS Proton peran layanan untuk penyediaan menggunakan CloudFormation.

Konsol Manajemen AWS

Gunakan konsol buat lingkungan di satu akun dan ketentuan di akun lain.

1. Di akun lingkungan, buat koneksi akun lingkungan, dan gunakan untuk mengirim permintaan untuk terhubung ke akun manajemen.

   a. Di AWS Proton konsol, pilih Koneksi akun Lingkungan di panel navigasi.

   b. Di halaman Koneksi akun Lingkungan, pilih Permintaan untuk terhubung.

   > **Note**
   >
   > Verifikasi bahwa ID akun yang tercantum dalam judul halaman koneksi akun Lingkungan cocok dengan ID akun lingkungan yang telah Anda identifikasi sebelumnya.

   c. Di halaman Permintaan untuk menghubungkan, di bagian Peran lingkungan, pilih Peran layanan yang ada dan nama peran layanan yang Anda buat untuk lingkungan.

   d. Di bagian Connect to management account, masukkan ID akun Manajemen dan nama Lingkungan untuk AWS Proton lingkungan Anda. Salin nama untuk digunakan nanti.

   e. Pilih Permintaan untuk terhubung di sudut kanan bawah halaman.

   f. Permintaan Anda ditampilkan sebagai tertunda dalam koneksi Lingkungan yang dikirim ke tabel akun manajemen dan modal menunjukkan cara menerima permintaan dari akun manajemen.

2. Di akun manajemen, terima permintaan untuk terhubung dari akun lingkungan.

   a. Masuk ke akun manajemen Anda dan pilih Koneksi akun Lingkungan di AWS Proton konsol.

   b. Di halaman Koneksi akun Lingkungan, di tabel Permintaan koneksi akun Lingkungan, pilih koneksi akun lingkungan dengan ID akun lingkungan yang cocok dengan ID akun lingkungan yang telah diidentifikasi sebelumnya.

   > **Note**
   >
   > Verifikasi bahwa ID akun yang tercantum dalam judul halaman koneksi akun Lingkungan cocok dengan ID akun manajemen Anda yang telah diidentifikasi sebelumnya.

c. Pilih Terima. Status berubah dari PENDING ke CONNECTED.

3. Di akun manajemen, buat lingkungan.

a. Di panel navigasi, pilih Template lingkungan.

b. Di halaman Template lingkungan, pilih Buat template lingkungan.

c. Di halaman Pilih template lingkungan, pilih template lingkungan.

d. Di halaman Konfigurasi lingkungan, di bagian Penyediaan, pilih penyediaan AWS terkelola.

e. Di bagian Akun Deployment, pilih AWS Akun lain;.

f. Di bagian Detail lingkungan, pilih koneksi akun Lingkungan dan nama Lingkungan Anda.

g. Pilih Berikutnya.

h. Isi formulir dan pilih Berikutnya sampai Anda mencapai halaman Review and Create.

i. Tinjau dan pilih Buat lingkungan.

AWS CLI

Gunakan AWS CLI untuk membuat lingkungan di satu akun dan ketentuan di akun lain.

Di akun lingkungan, buat koneksi akun lingkungan dan minta untuk terhubung dengan menjalankan perintah berikut.

```
$ aws proton create-environment-account-connection \
    --environment-name "simple-env-connected" \
    --role-arn "arn:aws:iam::222222222222:role/service-role/env-account-proton-service-role" \
    --management-account-id "111111111111"
```

Respons:

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:region-id:222222222222:environment-account-connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:13:50.847000+00:00",
```

```
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role",
        "status": "PENDING"
    }
}
```

Di akun manajemen, terima permintaan koneksi akun lingkungan dengan menjalankan perintah berikut.

```
$ aws proton accept-environment-account-connection \
    --id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

Respons:

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:region-id:222222222222:environment-account-
connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:15:33.486000+00:00",
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role",
        "status": "CONNECTED"
    }
}
```

Lihat koneksi akun lingkungan Anda dengan menjalankan perintah berikut.

```
$ aws proton get-environment-account-connection \
    --id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

Respons:

```
{
```

```
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:region-id:222222222222:environment-account-
connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:15:33.486000+00:00",
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role",
        "status": "CONNECTED"
    }
}
```

Di akun manajemen, buat lingkungan dengan menjalankan perintah berikut.

```
$ aws proton create-environment \
    --name "simple-env-connected" \
    --template-name simple-env-template \
    --template-major-version "1" \
    --template-minor-version "1" \
    --spec "file://simple-env-template/specs/original.yaml" \
    --environment-account-connection-id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

Respons:

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:111111111111:environment/simple-env-
connected",
        "createdAt": "2021-04-28T23:02:57.944000+00:00",
        "deploymentStatus": "IN_PROGRESS",
        "environmentAccountConnectionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
        "lastDeploymentAttemptedAt": "2021-04-28T23:02:57.944000+00:00",
        "name": "simple-env-connected",
        "templateName": "simple-env-template"
    }
}
```

Membuat dan menyediakan lingkungan menggunakan penyediaan yang dikelola sendiri

Saat Anda menggunakan penyediaan yang dikelola sendiri, AWS Proton mengirimkan permintaan tarik penyediaan ke repositori tertaut dengan infrastruktur penyediaan Anda sendiri. Permintaan tarik memulai alur kerja Anda sendiri, yang memanggil AWS layanan; untuk menyediakan infrastruktur.

Pertimbangan penyediaan yang dikelola sendiri:

- Sebelum Anda membuat lingkungan, siapkan direktori sumber daya repositori untuk penyediaan yang dikelola sendiri. Untuk informasi selengkapnya, lihat AWS Proton infrastruktur sebagai file kode.
- Setelah Anda membuat lingkungan, AWS Proton menunggu untuk menerima pemberitahuan asinkron mengenai status penyediaan infrastruktur Anda. Kode penyediaan Anda harus menggunakan AWS Proton `NotifyResourceStateChange` API untuk mengirim notifikasi asinkron ini. AWS Proton

Anda dapat menggunakan penyediaan yang dikelola sendiri di konsol atau dengan. AWS CLI Contoh berikut menunjukkan bagaimana Anda dapat menggunakan penyediaan yang dikelola sendiri dengan Terraform.

Konsol Manajemen AWS

Gunakan konsol untuk membuat lingkungan Terraform menggunakan penyediaan yang dikelola sendiri.

1. Di AWS Proton konsol, pilih Lingkungan.
2. Pilih Buat lingkungan.
3. Di halaman Pilih templat lingkungan, pilih templat Terraform dan pilih Konfigurasi.
4. Di halaman Konfigurasi lingkungan, di bagian Penyediaan, pilih Penyediaan yang dikelola sendiri.
5. Di bagian Rincian repositori penyediaan:
    a. Jika Anda belum menautkan repositori penyediaan Anda, pilih Repositori baru AWS Proton, pilih salah satu penyedia repositori, dan kemudian, untuk CodeStar koneksi, pilih salah satu koneksi Anda.

> **Note**
>
> Jika Anda belum memiliki koneksi ke akun penyedia repositori yang relevan, pilih Tambahkan koneksi baru CodeStar . Kemudian, buat koneksi, lalu pilih tombol refresh di sebelah menu CodeStar koneksi. Anda sekarang harus dapat memilih koneksi baru Anda di menu.

Jika Anda sudah menautkan repositori Anda AWS Proton, pilih Repositori yang ada.

b. Untuk nama Repositori, pilih repositori. Menu tarik-turun menunjukkan repositori tertaut untuk repositori yang ada atau daftar repositori di akun penyedia untuk repositori Baru.

c. Untuk nama Branch, pilih salah satu cabang repositori.

6. Di bagian Pengaturan lingkungan, masukkan nama Lingkungan.
7. (Opsional) Masukkan deskripsi untuk lingkungan.
8. (Opsional) Di bagian Tag, pilih Tambahkan tag baru dan masukkan kunci dan nilai untuk membuat tag terkelola pelanggan.
9. Pilih Berikutnya.
10. Di halaman Konfigurasi pengaturan khusus lingkungan, Anda harus memasukkan nilai untuk `required` parameter. Anda dapat memasukkan nilai untuk `optional` parameter atau menggunakan default saat diberikan.
11. Pilih Berikutnya dan tinjau masukan Anda.
12. Pilih Buat untuk mengirim permintaan tarik.

    - Jika Anda menyetujui permintaan tarik, penerapan sedang berlangsung.
    - Jika Anda menolak permintaan tarik, pembuatan lingkungan dibatalkan.
    - Jika waktu permintaan tarik habis, pembuatan lingkungan tidak selesai.

13. Lihat detail dan status lingkungan, serta tag AWS terkelola dan tag terkelola pelanggan untuk lingkungan Anda.
14. Pada panel navigasi, pilih Lingkungan.

    Halaman baru menampilkan daftar lingkungan Anda bersama dengan status dan detail lingkungan lainnya.

AWS CLI

Saat Anda membuat lingkungan menggunakan penyediaan yang dikelola sendiri, Anda menambahkan `provisioningRepository` parameter dan menghilangkan parameter dan parameter. `ProtonServiceRoleArn` `environmentAccountConnectionId`

Gunakan AWS CLI untuk membuat lingkungan Terraform dengan penyediaan yang dikelola sendiri.

1. Buat lingkungan dan kirim permintaan tarik ke repositori untuk ditinjau dan disetujui.

   Contoh berikutnya menunjukkan file spesifikasi YAML diformat yang mendefinisikan nilai untuk dua input berdasarkan file skema template lingkungan. Anda dapat menggunakan `get-environment-template-minor-version` perintah untuk melihat skema template lingkungan.

   Spesifikasi:

   ```
   proton: EnvironmentSpec
   spec:
     ssm_parameter_value: "test"
   ```

   Buat lingkungan dengan menjalankan perintah berikut.

   ```
   $ aws proton create-environment \
       --name "pr-environment" \
       --template-name "pr-env-template" \
       --template-major-version "1" \
       --provisioning-repository="branch=main,name=myrepos/env-
   repo,provider=GITHUB" \
       --spec "file://env-spec.yaml"
   ```

   Tanggapan: >

   ```
   {
       "environment": {
           "arn": "arn:aws:proton:region-id:123456789012:environment/pr-
   environment",
           "createdAt": "2021-11-18T17:06:58.679000+00:00",
           "deploymentStatus": "IN_PROGRESS",
           "lastDeploymentAttemptedAt": "2021-11-18T17:06:58.679000+00:00",
   ```

```
        "name": "pr-environment",
        "provisioningRepository": {
            "arn": "arn:aws:proton:region-id:123456789012:repository/
github:myrepos/env-repo",
            "branch": "main",
            "name": "myrepos/env-repo",
            "provider": "GITHUB"
        },
        "templateName": "pr-env-template"
    }
```

2. Tinjau permintaan.

   - Jika Anda menyetujui permintaan, penyediaan sedang berlangsung.
   - Jika Anda menolak permintaan, pembuatan lingkungan dibatalkan.
   - Jika waktu permintaan tarik habis, pembuatan lingkungan tidak selesai.

3. Berikan status penyediaan secara asinkron ke. AWS Proton Contoh berikut memberitahukan tentang AWS Proton penyediaan yang berhasil.

```
$ aws proton notify-resource-deployment-status-change \
    --resource-arn "arn:aws:proton:region-id:123456789012:environment/pr-
environment" \
    --status "SUCCEEDED"
```

Lihat data lingkungan

Anda dapat melihat data detail lingkungan menggunakan AWS Proton konsol atau file AWS CLI.

Konsol Manajemen AWS

Anda dapat melihat daftar lingkungan dengan detail dan lingkungan individual dengan data detail menggunakan AWS Proton konsol.

1. Untuk melihat daftar lingkungan Anda, pilih Lingkungan di panel navigasi.
2. Untuk melihat data detail, pilih nama lingkungan.

   Lihat data detail lingkungan Anda.

AWS CLI

Gunakan detail lingkungan AWS CLI get atau list.

Jalankan perintah berikut:

```
$ aws proton get-environment \
    --name "MySimpleEnv"
```

Respons:

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/MySimpleEnv",
        "createdAt": "2020-11-11T23:03:05.405000+00:00",
        "deploymentStatus": "SUCCEEDED",
        "lastDeploymentAttemptedAt": "2020-11-11T23:03:05.405000+00:00",
        "lastDeploymentSucceededAt": "2020-11-11T23:03:05.405000+00:00",
        "name": "MySimpleEnv",
        "protonServiceRoleArn": "arn:aws:iam::123456789012:role/ProtonServiceRole",
        "spec": "proton: EnvironmentSpec\nspec:\n  my_sample_input: \"the first\"\n
 my_other_sample_input: \"the second\"\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "0",
        "templateName": "simple-env"
    }
}
```

Perbarui lingkungan

Jika AWS Proton lingkungan dikaitkan dengan koneksi akun lingkungan, jangan perbarui atau sertakan `protonServiceRoleArn` parameter untuk memperbarui atau menyambung ke koneksi akun lingkungan.

Anda hanya dapat memperbarui ke koneksi akun lingkungan baru jika kedua hal berikut ini benar:

- Koneksi akun lingkungan dibuat di akun lingkungan yang sama dengan koneksi akun lingkungan saat ini dibuat.
- > Koneksi akun lingkungan dikaitkan dengan lingkungan saat ini.

Jika lingkungan tidak terkait dengan koneksi akun lingkungan, jangan perbarui atau sertakan `environmentAccountConnectionId` parameternya.

Anda dapat memperbarui `protonServiceRoleArn` parameter `environmentAccountConnectionId` atau nilai. Anda tidak dapat memperbarui keduanya.

Jika lingkungan Anda menggunakan penyediaan yang dikelola sendiri, jangan perbarui `provisioning-repository` parameter dan hilangkan parameter dan. `environmentAccountConnectionId` `protonServiceRoleArn`

Ada empat mode untuk memperbarui lingkungan seperti yang dijelaskan dalam daftar berikut. Saat menggunakan AWS CLI, `deployment-type` bidang mendefinisikan mode. Saat menggunakan konsol, mode ini dipetakan ke Edit, Perbarui, Perbarui minor, dan Perbarui tindakan utama yang diturunkan dari Tindakan.

`NONE`

Dalam mode ini, penerapan tidak terjadi. Hanya parameter metadata yang diminta yang diperbarui.

`CURRENT_VERSION`

Dalam mode ini, lingkungan diterapkan dan diperbarui dengan spesifikasi baru yang Anda berikan. Hanya parameter yang diminta yang diperbarui. Jangan sertakan parameter versi minor atau mayor saat Anda menggunakan ini`deployment-type`.

`MINOR_VERSION`

Dalam mode ini, lingkungan digunakan dan diperbarui dengan versi minor yang diterbitkan dan direkomendasikan (terbaru) dari versi utama saat ini yang digunakan secara default. Anda juga dapat menentukan versi minor yang berbeda dari versi utama saat ini yang digunakan.

`MAJOR_VERSION`

Dalam mode ini, lingkungan digunakan dan diperbarui dengan versi mayor dan minor yang diterbitkan, direkomendasikan (terbaru) dari template saat ini secara default. Anda juga dapat menentukan versi mayor yang berbeda yang lebih tinggi dari versi utama yang digunakan dan versi minor (opsional).

Topik

- Memperbarui lingkungan penyediaan AWS terkelola
- Memperbarui lingkungan penyediaan yang dikelola sendiri
- Membatalkan penerapan lingkungan yang sedang berlangsung

Memperbarui lingkungan penyediaan AWS terkelola

Penyediaan standar hanya didukung oleh lingkungan yang menyediakan. CloudFormation

Gunakan konsol atau AWS CLI untuk memperbarui lingkungan Anda.

Konsol Manajemen AWS

Perbarui lingkungan menggunakan konsol seperti yang ditunjukkan pada langkah-langkah berikut.

1. Pilih 1 dari 2 langkah berikut.
   a. Dalam daftar lingkungan.
      i. Di AWS Proton konsol, pilih Lingkungan.
      ii. Dalam daftar lingkungan, pilih tombol radio di sebelah kiri lingkungan yang ingin Anda perbarui.
   b. Di halaman detail lingkungan konsol.
      i. Di AWS Proton konsol, pilih Lingkungan.
      ii. Dalam daftar lingkungan, pilih nama lingkungan yang ingin Anda perbarui.
2. Pilih 1 dari 4 langkah berikutnya untuk memperbarui lingkungan Anda.
   a. Untuk melakukan pengeditan yang tidak memerlukan penerapan lingkungan.
      i. Misalnya, untuk mengubah deskripsi.

         Pilih Edit.
      ii. Isi formulir dan pilih Berikutnya.
      iii. Tinjau hasil edit Anda dan pilih Perbarui.
   b. Untuk membuat pembaruan pada input metadata saja.
      i. Pilih Tindakan dan kemudian Perbarui.

ii. Isi formulir dan pilih Edit.

iii. Isi formulir dan pilih Berikutnya hingga Anda mencapai halaman Ulasan.

iv. Tinjau pembaruan Anda dan pilih Perbarui.

c. Untuk membuat pembaruan ke versi minor baru dari template lingkungannya.

i. Pilih Tindakan dan kemudian Perbarui minor.

ii. Isi formulir dan pilih Berikutnya.

iii. Isi formulir dan pilih Berikutnya hingga Anda mencapai halaman Ulasan.

iv. Tinjau pembaruan Anda dan pilih Perbarui.

d. Untuk membuat pembaruan ke versi utama baru dari template lingkungannya.

i. Pilih Tindakan dan kemudian Perbarui mayor.

ii. Isi formulir dan pilih Berikutnya.

iii. Isi formulir dan pilih Berikutnya hingga Anda mencapai halaman Ulasan.

iv. Tinjau pembaruan Anda dan pilih Perbarui.

AWS CLI

Gunakan AWS Proton AWS CLI untuk memperbarui lingkungan ke versi minor baru.

Jalankan perintah berikut untuk memperbarui lingkungan Anda:

```
$ aws proton update-environment \
        --name "MySimpleEnv" \
        --deployment-type "MINOR_VERSION" \
        --template-major-version "1" \
        --template-minor-version "1" \
        --proton-service-role-arn arn:aws:iam::123456789012:role/service-
role/ProtonServiceRole \
        --spec "file:///spec.yaml"
```

Respons:

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/MySimpleEnv",
        "createdAt": "2021-04-02T17:29:55.472000+00:00",
        "deploymentStatus": "IN_PROGRESS",
```

```
        "lastDeploymentAttemptedAt": "2021-04-02T17:48:26.307000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T17:29:55.472000+00:00",
        "name": "MySimpleEnv",
        "protonServiceRoleArn": "arn:aws:iam::123456789012:role/service-role/
ProtonServiceRole",
        "templateMajorVersion": "1",
        "templateMinorVersion": "0",
        "templateName": "simple-env"
    }
}
```

Jalankan perintah berikut untuk mendapatkan dan mengkonfirmasi status:

```
$ aws proton get-environment \
        --name "MySimpleEnv"
```

Respons:

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/MySimpleEnv",
        "createdAt": "2021-04-02T17:29:55.472000+00:00",
        "deploymentStatus": "SUCCEEDED",
        "environmentName": "MySimpleEnv",
        "lastDeploymentAttemptedAt": "2021-04-02T17:48:26.307000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T17:48:26.307000+00:00",
        "protonServiceRoleArn": "arn:aws:iam::123456789012:role/service-role/
ProtonServiceRole",
        "spec": "proton: EnvironmentSpec\n\nspec:\n  my_sample_input: hello\n
 my_other_sample_input: everybody\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
        "templateName": "simple-env"
    }
}
```

Memperbarui lingkungan penyediaan yang dikelola sendiri

Penyediaan yang dikelola sendiri hanya didukung oleh lingkungan yang menyediakan Terraform.

Gunakan konsol atau AWS CLI untuk memperbarui lingkungan Anda.

Konsol Manajemen AWS

Perbarui lingkungan menggunakan konsol seperti yang ditunjukkan pada langkah-langkah berikut.

1. Pilih 1 dari 2 langkah berikut.
   a. Dalam daftar lingkungan.
      i. Di AWS Proton konsol, pilih Lingkungan.
      ii. Dalam daftar lingkungan, pilih tombol radio di sebelah kiri templat lingkungan yang ingin Anda perbarui.
   b. Di halaman detail lingkungan konsol.
      i. Di AWS Proton konsol, pilih Lingkungan.
      ii. Dalam daftar lingkungan, pilih nama lingkungan yang ingin Anda perbarui.
2. Pilih 1 dari 4 langkah berikutnya untuk memperbarui lingkungan Anda.
   a. Untuk melakukan pengeditan yang tidak memerlukan penerapan lingkungan.
      i. Misalnya, untuk mengubah deskripsi.

         Pilih Edit.
      ii. Isi formulir dan pilih Berikutnya.
      iii. Tinjau hasil edit Anda dan pilih Perbarui.
   b. Untuk membuat pembaruan pada input metadata saja.
      i. Pilih Tindakan dan kemudian Perbarui.
      ii. Isi formulir dan pilih Edit.
      iii. Isi formulir dan pilih Berikutnya hingga Anda mencapai halaman Ulasan.
      iv. Tinjau pembaruan Anda dan pilih Perbarui.
   c. Untuk membuat pembaruan ke versi minor baru dari template lingkungannya.
      i. Pilih Tindakan dan kemudian Perbarui minor.
      ii. Isi formulir dan pilih Berikutnya.
      iii. Isi formulir dan pilih Berikutnya hingga Anda mencapai halaman Ulasan.
      iv. Tinjau pembaruan Anda dan pilih Perbarui.

d. Untuk membuat pembaruan ke versi utama baru dari template lingkungannya.

   i. Pilih Tindakan dan kemudian Perbarui mayor.

   ii. Isi formulir dan pilih Berikutnya.

   iii. Isi formulir dan pilih Berikutnya hingga Anda mencapai halaman Ulasan.

   iv. Tinjau pembaruan Anda dan pilih Perbarui.

AWS CLI

Gunakan AWS CLI untuk memperbarui lingkungan Terraform ke versi minor baru dengan penyediaan yang dikelola sendiri.

1. Jalankan perintah berikut untuk memperbarui lingkungan Anda:

```
$ aws proton update-environment \
    --name "pr-environment" \
    --deployment-type "MINOR_VERSION" \
    --template-major-version "1" \
    --template-minor-version "1" \
    --provisioning-repository "branch=main,name=myrepos/env-
repo,provider=GITHUB" \
    --spec "file://env-spec-mod.yaml"
```

Respons:

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/pr-
environment",
        "createdAt": "2021-11-18T21:09:15.745000+00:00",
        "deploymentStatus": "IN_PROGRESS",
        "lastDeploymentAttemptedAt": "2021-11-18T21:25:41.998000+00:00",
        "lastDeploymentSucceededAt": "2021-11-18T21:09:15.745000+00:00",
        "name": "pr-environment",
        "provisioningRepository": {
            "arn": "arn:aws:proton:region-id:123456789012:repository/
github:myrepos/env-repo",
            "branch": "main",
            "name": "myrepos/env-repo",
            "provider": "GITHUB"
```

```
        },
        "templateMajorVersion": "1",
        "templateMinorVersion": "0",
        "templateName": "pr-env-template"
    }
}
```

2. Jalankan perintah berikut untuk mendapatkan dan mengkonfirmasi status:

```
$ aws proton get-environment \
    --name "pr-environment"
```

Respons:

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/pr-
environment",
        "createdAt": "2021-11-18T21:09:15.745000+00:00",
        "deploymentStatus": "SUCCEEDED",
        "lastDeploymentAttemptedAt": "2021-11-18T21:25:41.998000+00:00",
        "lastDeploymentSucceededAt": "2021-11-18T21:25:41.998000+00:00",
        "name": "pr-environment",
        "provisioningRepository": {
            "arn": "arn:aws:proton:region-id:123456789012:repository/
github:myrepos/env-repo",
            "branch": "main",
            "name": "myrepos/env-repo",
            "provider": "GITHUB"
        },
        "spec": "proton: EnvironmentSpec\nspec:\n   ssm_parameter_value: \"test
\"\n ssm_another_parameter_value: \"update\"\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
        "templateName": "pr-env-template"
    }
}
```

3. Tinjau permintaan tarik yang dikirim oleh AWS Proton.

   - Jika Anda menyetujui permintaan, penyediaan sedang berlangsung.
   - Jika Anda menolak permintaan, pembuatan lingkungan dibatalkan.

- Jika waktu permintaan tarik habis, pembuatan lingkungan tidak selesai.

4. Berikan status penyediaan ke. AWS Proton

```
$ aws proton notify-resource-deployment-status-change \
    --resource-arn "arn:aws:proton:region-id:123456789012:environment/pr-environment" \
    --status "SUCCEEDED"
```

Membatalkan penerapan lingkungan yang sedang berlangsung

Anda dapat mencoba membatalkan penerapan pembaruan lingkungan jika `deploymentStatus` ada di`IN_PROGRESS`. AWS Proton upaya untuk membatalkan penyebaran. Pembatalan yang berhasil tidak dijamin.

Saat Anda membatalkan penerapan pembaruan, AWS Proton upaya untuk membatalkan penerapan seperti yang tercantum dalam langkah-langkah berikut.

Dengan AWS-managed provisioning, AWS Proton lakukan hal berikut:

- Menetapkan status penerapan ke`CANCELLING`.
- Menghentikan penerapan yang sedang berlangsung dan menghapus sumber daya baru apa pun yang dibuat oleh penerapan saat. `IN_PROGRESS`
- Menetapkan status penerapan ke`CANCELLED`.
- Mengembalikan status sumber daya ke keadaan sebelum penerapan dimulai.

Dengan penyediaan yang dikelola sendiri, AWS Proton lakukan hal berikut:

- Mencoba menutup permintaan tarik untuk mencegah penggabungan perubahan ke repositori Anda.
- Menyetel status penerapan ke `CANCELLED` jika permintaan tarik berhasil ditutup.

Untuk petunjuk tentang cara membatalkan penerapan lingkungan, lihat CancelEnvironmentDeploymentdi Referensi AWS Proton API.

Anda dapat menggunakan konsol atau CLI untuk membatalkan lingkungan yang sedang berlangsung.

Konsol Manajemen AWS

Gunakan konsol untuk membatalkan penerapan pembaruan lingkungan seperti yang ditunjukkan pada langkah-langkah berikut.

1. Di AWS Proton konsol, pilih Lingkungan di panel navigasi.
2. Dalam daftar lingkungan, pilih nama lingkungan dengan pembaruan penyebaran yang ingin Anda batalkan.
3. Jika status penyebaran pembaruan Anda sedang berlangsung, di halaman detail lingkungan, pilih Tindakan, lalu Batalkan penerapan.
4. Modal meminta Anda untuk mengonfirmasi bahwa Anda ingin membatalkan. Pilih Batalkan penerapan.
5. Status penyebaran pembaruan Anda diatur ke Membatalkan dan kemudian Dibatalkan untuk menyelesaikan pembatalan.

AWS CLI

Gunakan AWS Proton AWS CLI untuk membatalkan penyebaran pembaruan lingkungan IN_PROGRESS ke versi minor baru 2.

Kondisi tunggu disertakan dalam template yang digunakan untuk contoh ini sehingga pembatalan dimulai sebelum penerapan pembaruan berhasil.

Jalankan perintah berikut untuk membatalkan pembaruan:

```
$ aws proton cancel-environment-deployment \
        --environment-name "MySimpleEnv"
```

Respons:

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/MySimpleEnv",
        "createdAt": "2021-04-02T17:29:55.472000+00:00",
        "deploymentStatus": "CANCELLING",
        "lastDeploymentAttemptedAt": "2021-04-02T18:15:10.243000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T17:48:26.307000+00:00",
        "name": "MySimpleEnv",
```

```
        "protonServiceRoleArn": "arn:aws:iam::123456789012:role/service-role/
ProtonServiceRole",
        "spec": "proton: EnvironmentSpec\n\nspec:\n  my_sample_input: hello\n
 my_other_sample_input: everybody\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
        "templateName": "simple-env"
    }
}
```

Jalankan perintah berikut untuk mendapatkan dan mengonfirmasi status:”

```
$ aws proton get-environment \
        --name "MySimpleEnv"
```

Respons:

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/MySimpleEnv",
        "createdAt": "2021-04-02T17:29:55.472000+00:00",
        "deploymentStatus": "CANCELLED",
        "deploymentStatusMessage": "User initiated cancellation.",
        "lastDeploymentAttemptedAt": "2021-04-02T18:15:10.243000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T17:48:26.307000+00:00",
        "name": "MySimpleEnv",
        "protonServiceRoleArn": "arn:aws:iam::123456789012:role/service-role/
ProtonServiceRole",
        "spec": "proton: EnvironmentSpec\n\nspec:\n  my_sample_input: hello\n
 my_other_sample_input: everybody\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
        "templateName": "simple-env"
    }
}
```

Hapus lingkungan

Anda dapat menghapus AWS Proton lingkungan dengan menggunakan AWS Proton konsol atau file AWS CLI.

> **Note**
>
> Anda tidak dapat menghapus lingkungan yang memiliki komponen terkait. Untuk menghapus lingkungan seperti itu, Anda harus terlebih dahulu menghapus semua komponen yang berjalan di lingkungan. Untuk informasi selengkapnya tentang komponen, lihatKomponen-komponen.

Konsol Manajemen AWS

Hapus lingkungan menggunakan konsol seperti yang dijelaskan dalam dua opsi berikut.

Dalam daftar lingkungan.

1. Di AWS Proton konsol, pilih Lingkungan.
2. Dalam daftar lingkungan, pilih tombol radio di sebelah kiri lingkungan yang ingin Anda hapus.
3. Pilih Tindakan dan kemudian Hapus.
4. Modal meminta Anda untuk mengonfirmasi tindakan hapus.
5. Ikuti instruksi dan pilih Ya, hapus.

Di halaman detail lingkungan.

1. Di AWS Proton konsol, pilih Lingkungan.
2. Dalam daftar lingkungan, pilih nama lingkungan yang ingin Anda hapus.
3. Di halaman detail lingkungan, pilih Tindakan dan kemudian Hapus.
4. Modal meminta Anda untuk mengonfirmasi bahwa Anda ingin menghapus.
5. Ikuti instruksi dan pilih Ya, hapus.

AWS CLI

Gunakan AWS CLI untuk menghapus lingkungan.

Jangan menghapus lingkungan jika layanan atau instance layanan diterapkan ke lingkungan.

Jalankan perintah berikut:

```
$ aws proton delete-environment \
```

```
    --name "MySimpleEnv"
```

Respons:

```
{
    "environment": {
        "arn": "arn:aws:proton:region-id:123456789012:environment/MySimpleEnv",
        "createdAt": "2021-04-02T17:29:55.472000+00:00",
        "deploymentStatus": "DELETE_IN_PROGRESS",
        "lastDeploymentAttemptedAt": "2021-04-02T17:48:26.307000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T17:48:26.307000+00:00",
        "name": "MySimpleEnv",
        "protonServiceRoleArn": "arn:aws:iam::123456789012:role/ProtonServiceRole",
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
        "templateName": "simple-env"
    }
}
```

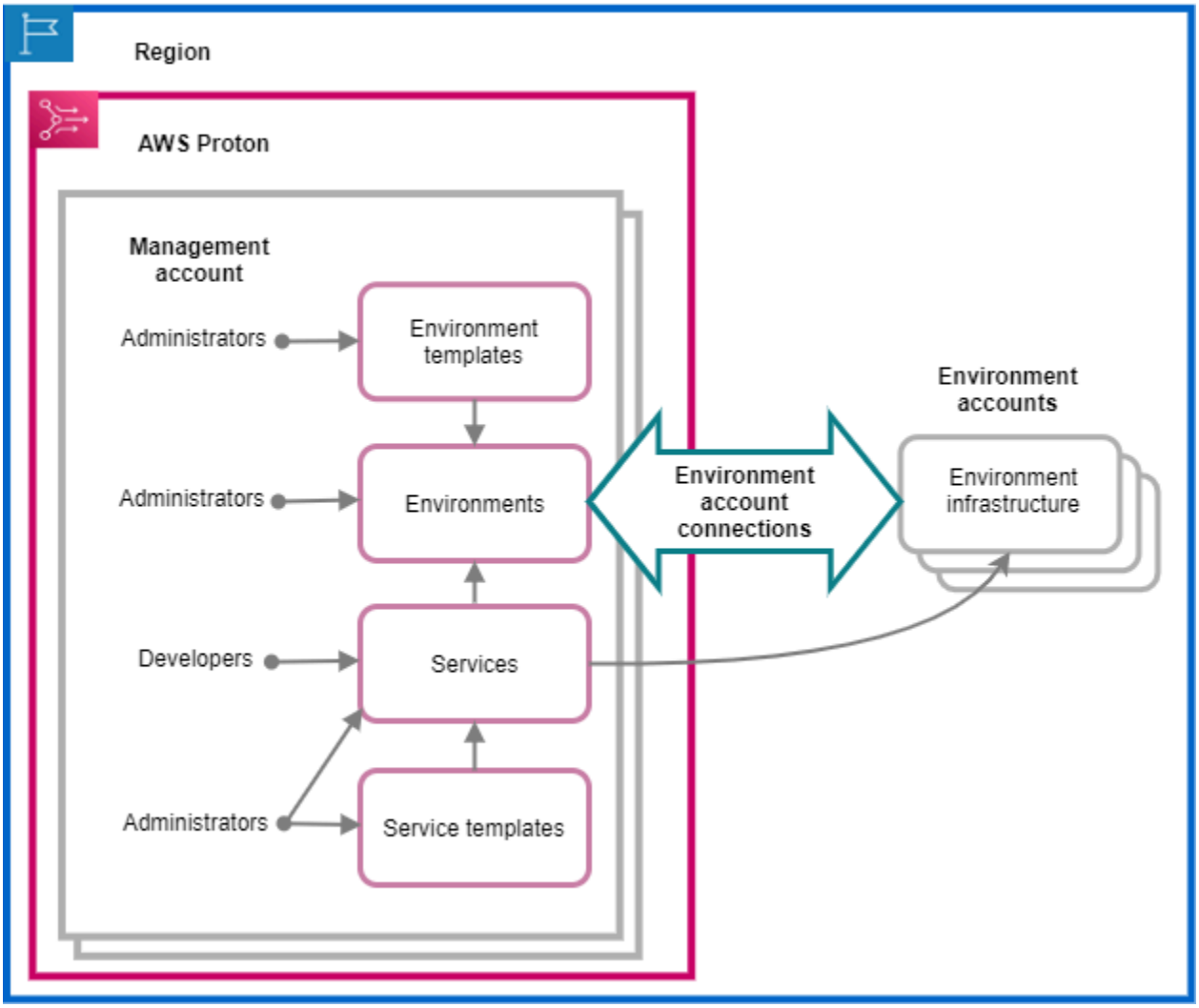
Koneksi akun lingkungan

Ikhtisar

Pelajari cara membuat dan mengelola AWS Proton lingkungan di satu akun dan menyediakan sumber daya infrastrukturnya di akun lain. Ini dapat membantu meningkatkan visibilitas dan efisiensi dalam skala besar. Koneksi akun lingkungan hanya mendukung penyediaan standar dengan CloudFormation infrastruktur sebagai kode.

> **Note**
>
> Informasi dalam topik ini relevan dengan lingkungan yang dikonfigurasi dengan penyediaan AWS terkelola. Dengan lingkungan yang dikonfigurasi dengan penyediaan yang dikelola sendiri, AWS Proton tidak secara langsung menyediakan infrastruktur Anda. Sebagai gantinya, ia mengirimkan pull requests (PRs) ke repositori Anda untuk penyediaan. Adalah tanggung jawab Anda untuk memastikan bahwa kode otomatisasi Anda mengasumsikan identitas dan peran yang tepat.
> Untuk informasi selengkapnya tentang metode penyediaan, lihat. the section called “Metode penyediaan”

Terminologi



Dengan koneksi akun AWS Proton lingkungan, Anda dapat membuat AWS Proton lingkungan dari satu akun dan menyediakan infrastrukturnya di akun lain.

Akun manajemen

Akun tunggal tempat Anda, sebagai administrator, membuat AWS Proton lingkungan yang menyediakan sumber daya infrastruktur di akun lingkungan lain.

Akun lingkungan

Akun tempat infrastruktur lingkungan disediakan, saat Anda membuat AWS Proton lingkungan di akun lain.

Koneksi akun lingkungan

Koneksi dua arah yang aman antara akun manajemen dan akun lingkungan. Ini mempertahankan otorisasi dan izin seperti yang dijelaskan lebih lanjut di bagian berikut.

Saat Anda membuat koneksi akun lingkungan di akun lingkungan di Wilayah tertentu, hanya akun manajemen di Wilayah yang sama yang dapat melihat dan menggunakan koneksi akun lingkungan. Ini berarti bahwa AWS Proton lingkungan yang dibuat dalam akun manajemen dan infrastruktur lingkungan yang disediakan dalam akun lingkungan harus berada di Wilayah yang sama.

Pertimbangan koneksi akun lingkungan

- Anda memerlukan koneksi akun lingkungan untuk setiap lingkungan yang ingin Anda sediakan di akun lingkungan.
- Untuk informasi tentang kuota koneksi akun lingkungan, lihatAWS Proton kuota.

Menandai

Di akun lingkungan, gunakan konsol atau AWS CLI untuk melihat dan mengelola koneksi akun lingkungan tag yang dikelola pelanggan. AWS tag terkelola tidak dibuat untuk koneksi akun lingkungan. Untuk informasi selengkapnya, lihat Penandaan.

Buat lingkungan di satu akun dan sediakan infrastrukturnya di akun lain

Untuk membuat dan menyediakan lingkungan dari satu akun manajemen, siapkan akun lingkungan untuk lingkungan yang ingin Anda buat.

Mulai di akun lingkungan dan buat koneksi.

Di akun lingkungan, buat peran AWS Proton layanan yang hanya mencakup izin yang diperlukan untuk menyediakan sumber daya infrastruktur lingkungan Anda. Untuk informasi selengkapnya, lihat AWS Proton peran layanan untuk penyediaan menggunakan CloudFormation.

Kemudian, buat dan kirim permintaan koneksi akun lingkungan ke akun manajemen Anda. Ketika permintaan diterima, AWS Proton dapat menggunakan peran IAM terkait yang memungkinkan penyediaan sumber daya lingkungan di akun lingkungan terkait.

Di akun manajemen, terima atau tolak koneksi akun lingkungan.

Di akun manajemen, terima atau tolak permintaan koneksi akun lingkungan. Anda tidak dapat menghapus koneksi akun lingkungan dari akun manajemen Anda.

Jika Anda menerima permintaan, AWS Proton dapat menggunakan peran IAM terkait yang mengizinkan penyediaan sumber daya di akun lingkungan terkait.

Sumber daya infrastruktur lingkungan disediakan di akun lingkungan terkait. Anda hanya dapat menggunakan AWS Proton APIs untuk mengakses dan mengelola lingkungan Anda dan sumber daya infrastrukturnya, dari akun manajemen Anda. Untuk informasi selengkapnya, lihat Buat lingkungan di satu akun dan ketentuan di akun lain dan Perbarui lingkungan.

Setelah menolak permintaan, Anda tidak dapat menerima atau menggunakan koneksi akun lingkungan yang ditolak.

> **Note**
>
> Anda tidak dapat menolak koneksi akun lingkungan yang terhubung ke lingkungan. Untuk menolak koneksi akun lingkungan, Anda harus terlebih dahulu menghapus lingkungan terkait.

Di akun lingkungan, akses sumber daya infrastruktur yang disediakan.

Di akun lingkungan, Anda dapat melihat dan mengakses sumber daya infrastruktur yang disediakan. Misalnya, Anda dapat menggunakan tindakan CloudFormation API untuk memantau dan membersihkan tumpukan jika diperlukan. Anda tidak dapat menggunakan tindakan AWS Proton API untuk mengakses atau mengelola AWS Proton lingkungan yang digunakan untuk menyediakan sumber daya infrastruktur.

Di akun lingkungan, Anda dapat menghapus koneksi akun lingkungan yang telah Anda buat di akun lingkungan. Anda tidak dapat menerima atau menolaknya. Jika Anda menghapus koneksi akun lingkungan yang digunakan oleh AWS Proton lingkungan, AWS Proton tidak akan dapat mengelola sumber daya infrastruktur lingkungan hingga koneksi lingkungan baru diterima untuk akun lingkungan dan lingkungan bernama. Anda bertanggung jawab untuk membersihkan sumber daya yang disediakan yang tetap tanpa koneksi lingkungan.

Gunakan konsol atau CLI untuk mengelola koneksi akun lingkungan

Anda dapat menggunakan konsol atau CLI untuk membuat dan mengelola koneksi akun lingkungan.

Konsol Manajemen AWS

Gunakan konsol untuk membuat koneksi akun lingkungan dan mengirim permintaan ke akun manajemen seperti yang ditunjukkan pada langkah berikutnya.

1. Tentukan nama untuk lingkungan yang Anda rencanakan untuk dibuat di akun manajemen Anda atau pilih nama lingkungan yang ada yang memerlukan koneksi akun lingkungan.
2. Di akun lingkungan, di AWS Proton konsol, pilih Koneksi akun Lingkungan di panel navigasi.
3. Di halaman Koneksi akun Lingkungan, pilih Permintaan untuk terhubung.

   > **Note**
   >
   > Verifikasi ID akun yang tercantum di judul halaman koneksi akun Lingkungan. Pastikan bahwa itu cocok dengan ID akun dari akun lingkungan yang Anda inginkan untuk disediakan oleh lingkungan bernama Anda.

4. Di halaman Permintaan untuk terhubung:

   a. Di bagian Connect to management account, masukkan ID akun Manajemen dan nama Lingkungan yang Anda masukkan pada langkah 1.

   b. Di bagian Peran lingkungan, pilih Peran layanan baru dan AWS Proton secara otomatis membuat peran baru untuk Anda. Atau, pilih Peran layanan yang ada dan nama peran layanan yang Anda buat sebelumnya.

      > **Note**
      >
      > Peran yang dibuat AWS Proton secara otomatis untuk Anda memiliki izin luas. Kami menyarankan Anda untuk memasukkan peran ke izin yang diperlukan untuk menyediakan sumber daya infrastruktur lingkungan Anda. Untuk informasi selengkapnya, lihat AWS Proton peran layanan untuk penyediaan menggunakan CloudFormation.

   c. (Opsional) Di bagian Tag, pilih Tambahkan tag baru untuk membuat tag terkelola pelanggan untuk koneksi akun lingkungan Anda.

   d. Pilih Permintaan untuk terhubung.

5. Permintaan Anda ditampilkan sebagai tertunda dalam koneksi Lingkungan yang dikirim ke tabel akun manajemen dan modal memungkinkan Anda mengetahui cara menerima permintaan dari akun manajemen.

Menerima atau menolak permintaan koneksi akun lingkungan.

1. Di akun manajemen, di AWS Proton konsol, pilih Koneksi akun Lingkungan di panel navigasi.
2. Di halaman Koneksi akun Lingkungan, di tabel permintaan koneksi akun Lingkungan, pilih permintaan koneksi lingkungan untuk menerima atau menolak.

> **Note**
>
> Verifikasi ID akun yang tercantum di judul halaman koneksi akun Lingkungan. Pastikan bahwa itu cocok dengan ID akun dari akun manajemen yang terkait dengan koneksi akun lingkungan untuk ditolak. Setelah Anda menolak koneksi akun lingkungan ini, Anda tidak dapat menerima atau menggunakan koneksi akun lingkungan yang ditolak.

3. Pilih Tolak atau Terima.
   - Jika Anda memilih Tolak, status akan berubah dari pending menjadi ditolak.
   - Jika Anda memilih Terima, status akan berubah dari pending menjadi terhubung.

Hapus koneksi akun lingkungan.

1. Di akun lingkungan, di AWS Proton konsol, pilih Koneksi akun Lingkungan di panel navigasi.

> **Note**
>
> Verifikasi ID akun yang tercantum di judul halaman koneksi akun Lingkungan. Pastikan bahwa itu cocok dengan ID akun dari akun manajemen yang terkait dengan koneksi akun lingkungan untuk ditolak. Setelah Anda menghapus koneksi akun lingkungan ini, tidak AWS Proton dapat mengelola sumber daya infrastruktur lingkungan di akun lingkungan. Itu hanya dapat mengelolanya setelah koneksi akun lingkungan baru untuk akun lingkungan dan lingkungan bernama diterima oleh akun manajemen.

2. Di halaman Koneksi akun Lingkungan, di bagian Permintaan terkirim untuk terhubung ke akun manajemen, pilih Hapus.
3. Modal meminta Anda untuk mengonfirmasi bahwa Anda ingin menghapus. Pilih Hapus.

AWS CLI

Tentukan nama untuk lingkungan yang Anda rencanakan untuk dibuat di akun manajemen Anda atau pilih nama lingkungan yang ada yang memerlukan koneksi akun lingkungan.

Buat koneksi akun lingkungan di akun lingkungan.

Jalankan perintah berikut:

```
$ aws proton create-environment-account-connection \
    --environment-name "simple-env-connected" \
    --role-arn "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role" \
    --management-account-id "111111111111"
```

Respons:

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:region-id:222222222222:environment-account-
connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:13:50.847000+00:00",
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role",
        "status": "PENDING"
    }
}
```

Terima atau tolak koneksi akun lingkungan di akun manajemen seperti yang ditunjukkan pada perintah dan respons berikut.

> **Note**
>
> Jika Anda menolak koneksi akun lingkungan ini, Anda tidak akan dapat menerima atau menggunakan koneksi akun lingkungan yang ditolak.

Jika Anda menentukan Tolak, status akan berubah dari pending menjadi ditolak.

Jika Anda menentukan Terima, status akan berubah dari pending menjadi terhubung.

Jalankan perintah berikut untuk menerima koneksi akun lingkungan:

```
$ aws proton accept-environment-account-connection \
    --id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

Respons:

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:region-id:222222222222:environment-account-
connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:15:33.486000+00:00",
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role",
        "status": "CONNECTED"
    }
}
```

Jalankan perintah berikut untuk menolak koneksi akun lingkungan:

```
$ aws proton reject-environment-account-connection \
    --id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

Respons:

```
{
```

```
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:us-east-1:222222222222:environment-account-
connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "status": "REJECTED",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-reject",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:13:50.847000+00:00",
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role"
    }
}
```

Lihat koneksi akun lingkungan. Anda bisa mendapatkan atau membuat daftar koneksi akun lingkungan.

Jalankan perintah get berikut:

```
$ aws proton get-environment-account-connection \
    --id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

Respons:

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:region-id:222222222222:environment-account-
connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:15:33.486000+00:00",
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role",
        "status": "CONNECTED"
    }
}
```

Hapus koneksi akun lingkungan di akun lingkungan.

> **Note**
>
> Jika Anda menghapus koneksi akun lingkungan ini, AWS Proton tidak akan dapat mengelola sumber daya infrastruktur lingkungan di akun lingkungan hingga koneksi lingkungan baru diterima untuk akun lingkungan dan lingkungan bernama. Anda bertanggung jawab untuk membersihkan sumber daya yang disediakan yang tetap tanpa koneksi lingkungan.

Jalankan perintah berikut:

```
$ aws proton delete-environment-account-connection \
    --id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

Respons:

```
{
    "environmentAccountConnection": {
        "arn": "arn:aws:proton:us-east-1:222222222222:environment-account-
connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "environmentAccountId": "222222222222",
        "environmentName": "simple-env-connected",
        "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "lastModifiedAt": "2021-04-28T23:13:50.847000+00:00",
        "managementAccountId": "111111111111",
        "requestedAt": "2021-04-28T23:13:50.847000+00:00",
        "roleArn": "arn:aws:iam::222222222222:role/service-role/env-account-proton-
service-role",
        "status": "CONNECTED"
    }
}
```

Lingkungan yang dikelola pelanggan

Dengan lingkungan yang dikelola pelanggan, Anda dapat menggunakan infrastruktur yang ada, seperti VPC, yang telah Anda gunakan sebagai lingkungan Anda. AWS Proton Saat menggunakan lingkungan yang dikelola pelanggan, Anda dapat menyediakan sumber daya bersama Anda sendiri di luar. AWS Proton Namun, Anda masih dapat mengizinkan AWS Proton untuk menggunakan output penyediaan yang relevan sebagai input untuk AWS Proton layanan saat digunakan. Jika output dapat

berubah, AWS Proton dapat menerima pembaruan. AWS Proton Namun, tidak dapat mengubah lingkungan secara langsung, karena penyediaan dikelola di luar. AWS Proton

Setelah lingkungan dibuat, Anda bertanggung jawab untuk menyediakan output yang sama dengan AWS Proton yang akan dibuat jika AWS Proton telah membuat lingkungan, seperti nama cluster Amazon ECS atau Amazon VPC. IDs

Dengan fungsi ini, Anda dapat menyebarkan dan memperbarui sumber daya AWS Proton layanan dari template AWS Proton layanan ke lingkungan ini. Namun, lingkungan itu sendiri tidak dimodifikasi melalui pembaruan template di AWS Proton. Anda bertanggung jawab untuk menjalankan pembaruan ke lingkungan dan memperbarui output tersebut. AWS Proton

Anda dapat memiliki beberapa lingkungan dalam satu akun yang merupakan campuran lingkungan yang AWS Proton dikelola dan dikelola pelanggan. Anda juga dapat menautkan akun kedua dan menggunakan AWS Proton templat di akun utama untuk menjalankan penerapan dan pembaruan ke lingkungan dan layanan di akun kedua yang ditautkan.

Cara menggunakan lingkungan yang dikelola pelanggan

Hal pertama yang perlu dilakukan administrator adalah mendaftarkan template lingkungan yang diimpor dan dikelola pelanggan. Jangan menyediakan manifes atau file infrastruktur dalam bundel template. Hanya menyediakan skema.

Skema di bawah ini menguraikan daftar output menggunakan format API terbuka dan mereplikasi output dari template. CloudFormation

> **Important**
>
> Hanya input string yang diizinkan untuk output.

Contoh berikut adalah cuplikan bagian output dari CloudFormation template untuk template Fargate yang sesuai.

```
Outputs:
  ClusterName:
    Description: The name of the ECS cluster
    Value: !Ref 'ECSCluster'
  ECSTaskExecutionRole:
    Description: The ARN of the ECS role
```

```
    Value: !GetAtt 'ECSTaskExecutionRole.Arn'
  VpcId:
    Description: The ID of the VPC that this stack is deployed in
    Value: !Ref 'VPC'
[...]
```

Skema untuk lingkungan AWS Proton impor yang sesuai mirip dengan yang berikut ini. Jangan berikan nilai default dalam skema.

```
schema:
  format:
    openapi: "3.0.0"
  environment_input_type: "EnvironmentOutput"
  types:
    EnvironmentOutput:
      type: object
      description: "Outputs of the environment"
      properties:
        ClusterName:
          type: string
          description: "The name of the ECS cluster"
        ECSTaskExecutionRole:
          type: string
          description: "The ARN of the ECS role"
        VpcId:
          type: string
          description: "The ID of the VPC that this stack is deployed in"
[...]
```

Pada saat mendaftarkan template, Anda menunjukkan bahwa template ini diimpor dan menyediakan lokasi bucket Amazon S3 untuk bundel. AWS Proton memvalidasi bahwa skema hanya berisi `environment_input_type` dan tidak ada parameter CloudFormation template sebelum menempatkan template dalam draf.

Anda memberikan yang berikut ini untuk membuat lingkungan yang diimpor.

- Peran IAM untuk digunakan saat membuat penerapan.
- Spesifikasi dengan nilai untuk output yang diperlukan.

Anda dapat menyediakan keduanya melalui konsol atau AWS CLI menggunakan proses yang mirip dengan penerapan lingkungan biasa.

CodeBuild pembuatan peran penyediaan

Alat Infrastruktur sebagai Kode (IAAC) seperti CloudFormation dan Terraform memerlukan izin untuk berbagai jenis sumber daya. AWS Misalnya, jika templat IAAC mendeklarasikan bucket Amazon S3, templat tersebut memerlukan izin untuk membuat, membaca, memperbarui, dan menghapus bucket Amazon S3. Ini dianggap sebagai praktik terbaik keamanan untuk membatasi peran ke izin minimal yang diperlukan. Mengingat luasnya AWS sumber daya, sulit untuk membuat kebijakan hak istimewa terkecil untuk templat IAAC, terutama ketika sumber daya yang dikelola oleh templat tersebut dapat berubah nanti. Misalnya, dalam pengeditan terbaru Anda ke template yang dikelola oleh AWS Proton, Anda menambahkan sumber daya database RDS.

Mengkonfigurasi izin yang tepat membantu membuat penerapan IAC Anda lancar. AWS Proton CodeBuild Penyediaan mengeksekusi perintah CLI yang disediakan pelanggan sewenang-wenang dalam proyek yang terletak di akun CodeBuild pelanggan. Biasanya, perintah ini membuat dan menghapus infrastruktur menggunakan alat Infrastructure as Code (IAAC) seperti. AWS CDK Ketika AWS sumber daya menyebarkan yang templatnya menggunakan CodeBuild Provisioning, AWS akan memulai build dalam CodeBuild proyek yang dikelola oleh. AWS Sebuah peran diteruskan ke CodeBuild, yang CodeBuild mengasumsikan untuk menjalankan perintah. Peran ini, yang disebut CodeBuild Peran Penyediaan, disediakan oleh pelanggan dan berisi izin yang diperlukan untuk menyediakan infrastruktur. Ini dimaksudkan untuk diasumsikan hanya oleh CodeBuild dan bahkan tidak AWS Proton dapat mengasumsikan itu.

Menciptakan peran

CodeBuild Peran Penyediaan dapat dibuat di konsol IAM atau di. AWS CLI Untuk membuatnya di AWS CLI:

```
aws iam create-role --role-name AWSProtonCodeBuildProvisioning --assume-role-policy-
document '{"Version": "2012-10-17",       "Statement":[{"Effect":"Allow","Principal":
{"Service":"codebuild.amazonaws.com"},"Action":"sts:AssumeRole"}]}'
aws iam attach-role-policy --role-name AWSProtonCodeBuildProvisioning --policy-arn
 arn:aws:iam::aws:policy/AWSProtonCodeBuildProvisioningBasicAccess
```

Ini juga melampirkan`AWSProtonCodeBuildProvisioningBasicAccess`, yang berisi izin minimal yang diperlukan oleh CodeBuild layanan untuk menjalankan build.

Jika Anda lebih suka menggunakan konsol, pastikan hal berikut saat Anda membuat peran:

1. Untuk entitas tepercaya, pilih AWS layanan lalu pilih CodeBuild.

2. Pada langkah Tambahkan izin, pilih `AWSProtonCodeBuildProvisioningBasicAccess` dan kebijakan lain yang ingin Anda lampirkan.

Akses Administrator

Jika Anda melampirkan `AdministratorAccess` kebijakan ke CodeBuild Peran Penyediaan, itu akan menjamin bahwa template IAAC tidak akan gagal karena kurangnya izin. Ini juga berarti bahwa siapa pun yang dapat membuat Template Lingkungan atau Template Layanan dapat melakukan tindakan tingkat administrator, bahkan jika pengguna tersebut bukan administrator. AWS Proton tidak merekomendasikan penggunaan `AdministatorAccess` dengan Peran CodeBuild Penyediaan. Jika Anda memutuskan untuk menggunakan `AdministratorAccess` CodeBuild Peran Penyediaan, lakukan di lingkungan kotak pasir.

Anda dapat membuat peran dengan `AdministratorAccess` di konsol IAM atau dengan menjalankan perintah ini:

```
aws iam create-role --role-name AWSProtonCodeBuildProvisioning --assume-role-policy-
document '{"Version": "2012-10-17",       "Statement":[{"Effect":"Allow","Principal":
{"Service":"codebuild.amazonaws.com"},"Action":"sts:AssumeRole"}]}'
aws iam attach-role-policy --role-name AWSProtonCodeBuildProvisioning --policy-arn
 arn:aws:iam::aws:policy/AdministratorAccess
```

Membuat Peran Cakupan Minimal

Jika Anda ingin membuat peran dengan izin minimum, ada beberapa pendekatan:

- Terapkan dengan izin admin, lalu cakup perannya. Kami merekomendasikan menggunakan IAM Access Analyzer.
- Gunakan kebijakan terkelola untuk memberikan akses ke layanan yang Anda rencanakan untuk digunakan.

AWS CDK

Jika Anda menggunakan AWS CDK with AWS Proton, dan Anda telah menjalankan `cdk bootstrap` di setiap akun/wilayah lingkungan, maka sudah ada peran untuk. `cdk deploy` Dalam hal ini, lampirkan kebijakan berikut ke CodeBuild Peran Penyediaan:

```
{
    "Action": "sts:AssumeRole",
```

```
    "Resource": [
        "arn:aws:iam::account-id:role/cdk-*-deploy-role-*",
        "arn:aws:iam::account-id:role/cdk-*-file-publishing-role-*"
    ],
    "Effect": "Allow"
}
```

VPC kustom

Jika Anda memutuskan untuk menjalankan CodeBuild VPC kustom, Anda memerlukan izin berikut dalam peran Anda: CodeBuild

```
{
    "Effect": "Allow",
    "Action": [
        "ec2:CreateNetworkInterface"
    ],
    "Resource": [
        "arn:aws:ec2:region:account-id:network-interface/*",
        "arn:aws:ec2:region:account-id:subnet/*",
        "arn:aws:ec2:region:account-id:security-group/*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "ec2:DeleteNetworkInterface"
    ],
    "Resource": [
        "arn:aws:ec2:region:account-id:*/*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeVpcs"
    ],
    "Resource": "*"
},
```

```
{
    "Effect": "Allow",
    "Action": [
        "ec2:CreateNetworkInterfacePermission"
    ],
    "Resource": "arn:aws:ec2:region:account-id:network-interface/*",
    "Condition": {
        "StringEquals": {
            "ec2:AuthorizedService": "codebuild.amazonaws.com"
        }
    }
}
```

Anda juga dapat menggunakan kebijakan `AmazonEC2FullAccess` terkelola, meskipun itu termasuk izin yang mungkin tidak Anda perlukan. Untuk melampirkan kebijakan terkelola menggunakan CLI:

```
aws iam create-role --role-name AWSProtonCodeBuildProvisioning --assume-role-policy-
document '{"Version": "2012-10-17",       "Statement":[{"Effect":"Allow","Principal":
{"Service":"codebuild.amazonaws.com"},"Action":"sts:AssumeRole"}]}'
aws iam attach-role-policy --role-name AWSProtonCodeBuildProvisioning --policy-arn
 arn:aws:iam::aws:policy/AdministratorAccess
```

AWS Proton layanan

AWS Proton Layanan adalah instantiasi template layanan, biasanya termasuk beberapa instance layanan dan pipeline. Sebuah instance AWS Proton layanan adalah instantiasi dari template layanan dalam lingkungan tertentu. Template layanan adalah definisi lengkap dari infrastruktur dan pipa layanan opsional untuk suatu AWS Proton layanan.

Setelah menerapkan instance layanan, Anda dapat memperbaruinya dengan mendorong kode sumber yang meminta CI/CD pipeline atau dengan memperbarui layanan ke versi baru template layanannya. AWS Proton meminta Anda ketika versi baru dari template layanannya tersedia sehingga Anda dapat memperbarui layanan Anda ke versi baru. Saat layanan Anda diperbarui, AWS Proton gunakan kembali instance layanan dan layanan.

Bab ini menunjukkan cara mengelola layanan dengan menggunakan operasi membuat, melihat, memperbarui, dan menghapus. Untuk informasi tambahan, lihat Referensi API AWS Proton Layanan.

Topik

- Membuat layanan
- Lihat data layanan
- Mengedit layanan
- Hapus layanan
- Lihat data contoh layanan
- Memperbarui instance layanan
- Memperbarui saluran layanan

Membuat layanan

Untuk menyebarkan aplikasi dengan AWS Proton, sebagai pengembang, Anda membuat layanan dan memberikan masukan berikut.

1. Nama template AWS Proton layanan yang diterbitkan oleh tim platform.
2. Sebuah nama untuk layanan.
3. Jumlah instance layanan yang ingin Anda terapkan.
4. Pilihan lingkungan yang ingin Anda gunakan.

5. Koneksi ke repositori kode Anda jika Anda menggunakan template layanan yang menyertakan pipeline layanan (opsional).

Apa yang ada dalam layanan?

Saat Anda membuat AWS Proton layanan, Anda dapat memilih dari dua jenis templat layanan yang berbeda:

- Template layanan yang menyertakan pipeline layanan (default).
- Template layanan yang tidak menyertakan pipeline layanan.

Anda harus membuat setidaknya satu contoh layanan ketika Anda membuat layanan Anda.

Sebuah instance layanan dan pipa opsional dikaitkan dengan layanan. Anda hanya dapat membuat atau menghapus pipeline dalam konteks tindakan membuat dan menghapus layanan. Untuk mempelajari cara menambah dan menghapus instance dari layanan, lihatMengedit layanan.

> **Note**
>
> Lingkungan Anda dikonfigurasi untuk penyediaan AWS- atau yang dikelola sendiri. AWS Proton menyediakan layanan di lingkungan menggunakan metode penyediaan yang sama seperti yang digunakan lingkungan. Pengembang yang membuat atau memperbarui instance layanan tidak melihat perbedaannya dan pengalaman mereka sama dalam kedua kasus tersebut.
> Untuk informasi selengkapnya tentang metode penyediaan, lihat. the section called “Metode penyediaan”

Templat layanan

Baik versi utama dan minor dari template layanan tersedia. Saat Anda menggunakan konsol, Anda memilih versi `Recommended` mayor dan minor terbaru dari template layanan. Bila Anda menggunakan AWS CLI dan Anda hanya menentukan versi utama dari template layanan, Anda secara implisit menentukan versi `Recommended` minor terbarunya.

Berikut ini menjelaskan perbedaan antara versi template mayor dan minor dan penggunaannya.

- Versi baru template menjadi `Recommended` segera setelah disetujui oleh anggota tim platform. Ini berarti bahwa layanan baru dibuat menggunakan versi itu, dan Anda diminta untuk memperbarui layanan yang ada ke versi baru.
- Melalui AWS Proton, tim platform dapat secara otomatis memperbarui instance layanan ke versi minor baru dari template layanan. Versi minor harus kompatibel ke belakang.
- Karena versi utama mengharuskan Anda untuk memberikan input baru sebagai bagian dari proses pembaruan, Anda perlu memperbarui layanan Anda ke versi utama dari template layanannya. Versi utama tidak kompatibel ke belakang.

Membuat layanan

Prosedur berikut menunjukkan cara menggunakan AWS Proton konsol atau AWS CLI membuat layanan dengan atau tanpa pipa layanan.

Konsol Manajemen AWS

Buat layanan seperti yang ditunjukkan pada langkah-langkah konsol berikut.

1. Di AWS Proton konsol, pilih Layanan.
2. Pilih Buat layanan.
3. Di halaman Pilih templat layanan, pilih templat dan pilih Konfigurasi.

   Jika Anda tidak ingin menggunakan pipeline yang diaktifkan, pilih templat yang ditandai dengan pipeline Excludes untuk layanan Anda.
4. Di halaman Konfigurasi Layanan, di bagian Pengaturan layanan, masukkan nama Layanan.
5. (Opsional) Masukkan deskripsi untuk layanan.
6. Di bagian Pengaturan Repositori Layanan:

   a. Untuk CodeStar Koneksi, pilih koneksi Anda dari daftar.

   b. Untuk ID Repositori, pilih nama repositori kode sumber Anda dari daftar.

   c. Untuk nama Branch, pilih nama cabang repositori kode sumber Anda dari daftar.
7. (Opsional) Di bagian Tag, pilih Tambahkan tag baru dan masukkan kunci dan nilai untuk membuat tag terkelola pelanggan.
8. Pilih Berikutnya.

9. Di halaman Konfigurasi pengaturan kustom, di bagian Instans layanan, di bagian Instans baru. Anda harus memasukkan nilai untuk `required` parameter. Anda dapat memasukkan nilai untuk `optional` parameter atau menggunakan default saat diberikan.
10. Di bagian input Pipeline, Anda harus memasukkan nilai untuk `required` parameter. Anda dapat memasukkan nilai untuk `optional` parameter atau menggunakan default saat diberikan.
11. Pilih Berikutnya dan tinjau masukan Anda.
12. Pilih Buat.

    Lihat detail dan status layanan, serta tag AWS terkelola dan tag terkelola pelanggan untuk layanan Anda.
13. Pada panel navigasi, silakan pilih Layanan.

    Halaman baru menampilkan daftar layanan Anda bersama dengan status dan detail layanan lainnya.

AWS CLI

Saat Anda menggunakan AWS CLI, Anda menentukan input layanan dalam `spec` file berformat YAMAL`.aws-proton/service.yaml`, yang terletak di direktori kode sumber Anda.

Anda dapat menggunakan `get-service-template-minor-version` perintah CLI untuk melihat skema yang diperlukan dan parameter opsional yang Anda berikan nilai dalam file spesifikasi Anda.

Jika Anda ingin menggunakan template layanan yang memilikipipelineProvisioning: "CUSTOMER_MANAGED", jangan sertakan `pipeline:` bagian dalam spesifikasi Anda dan jangan sertakan`-repository-connection-arn`,`-repository-id`, dan `-branch-name` parameter dalam `create-service` perintah Anda.

Buat layanan dengan pipeline layanan seperti yang ditunjukkan pada langkah-langkah CLI berikut.

1. Siapkan peran layanan untuk pipeline seperti yang ditunjukkan pada perintah contoh CLI berikut.

   Perintah:

   ```
   $ aws proton update-account-settings \
   ```

```
        --pipeline-service-role-arn "arn:aws:iam::123456789012:role/AWS
 ProtonServiceRole"
```

2. Daftar berikut menunjukkan contoh spesifikasi, berdasarkan skema template layanan, yang mencakup pipeline layanan dan input instance.

   Spesifikasi:

```
proton: ServiceSpec

pipeline:
  my_sample_pipeline_required_input: "hello"
  my_sample_pipeline_optional_input: "bye"

instances:
  - name: "acme-network-dev"
    environment: "ENV_NAME"
    spec:
      my_sample_service_instance_required_input: "hi"
      my_sample_service_instance_optional_input: "ho"
```

   Buat layanan dengan pipeline seperti yang ditunjukkan pada contoh perintah dan respons CLI berikut.

   Perintah:

```
$ aws proton create-service \
        --name "MySimpleService" \
        --branch-name "mainline" \
        --template-major-version "1" \
        --template-name "fargate-service" \
        --repository-connection-arn "arn:aws:codestar-connections:region-
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111" \
        --repository-id "myorg/myapp" \
        --spec "file://spec.yaml"
```

   Respons:

```
{
    "service": {
        "arn": "arn:aws:proton:region-id:123456789012:service/MySimpleService",
        "createdAt": "2020-11-18T19:50:27.460000+00:00",
```

```
        "lastModifiedAt": "2020-11-18T19:50:27.460000+00:00",
        "name": "MySimpleService",
        "repositoryConnectionArn": "arn:aws:codestar-connections:region-
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "repositoryId": "myorg/myapp",
        "status": "CREATE_IN_PROGRESS",
        "templateName": "fargate-service"
    }
}
```

Buat layanan tanpa pipeline layanan seperti yang ditunjukkan pada contoh perintah dan respons CLI berikut.

Berikut ini menunjukkan contoh spesifikasi yang tidak menyertakan input pipeline layanan.

Spesifikasi:

```
proton: ServiceSpec

instances:
  - name: "acme-network-dev"
    environment: "ENV_NAME"
    spec:
      my_sample_service_instance_required_input: "hi"
      my_sample_service_instance_optional_input: "ho"
```

Untuk membuat layanan tanpa pipeline layanan yang disediakan, Anda menyediakan jalur ke a **`spec.yaml`** dan Anda tidak menyertakan parameter repositori seperti yang ditunjukkan pada perintah dan respons contoh CLI berikut.

Perintah:

```
$ aws proton create-service \
        --name "MySimpleServiceNoPipeline" \
        --template-major-version "1" \
        --template-name "fargate-service" \
        --spec "file://spec-no-pipeline.yaml"
```

Respons:

```
{
```

```
    "service": {
        "arn": "arn:aws:proton:region-id:123456789012:service/
MySimpleServiceNoPipeline",
        "createdAt": "2020-11-18T19:50:27.460000+00:00",
        "lastModifiedAt": "2020-11-18T19:50:27.460000+00:00",
        "name": "MySimpleServiceNoPipeline",
        "status": "CREATE_IN_PROGRESS",
        "templateName": "fargate-service-no-pipeline"
    }
}
```

Lihat data layanan

Anda dapat melihat dan mencantumkan data detail layanan menggunakan AWS Proton konsol atau file AWS CLI.

Konsol Manajemen AWS

Daftar dan lihat detail layanan menggunakan AWS Proton konsol seperti yang ditunjukkan pada langkah-langkah berikut.

1. Untuk melihat daftar layanan Anda, pilih Layanan di panel navigasi.
2. Untuk melihat data detail, pilih nama layanan.

   Lihat data detail layanan Anda.

AWS CLI

Lihat detail layanan dengan pipeline layanan seperti yang ditunjukkan pada perintah dan respons contoh CLI berikut.

Perintah:

```
$ aws proton get-service \
    --name "simple-svc"
```

Respons:

```
{
    "service": {
```

```
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc",
        "branchName": "mainline",
        "createdAt": "2020-11-28T22:40:50.512000+00:00",
        "lastModifiedAt": "2020-11-28T22:44:51.207000+00:00",
        "name": "simple-svc",
        "pipeline": {
            "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/
pipeline/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "createdAt": "2020-11-28T22:40:50.512000+00:00",
            "deploymentStatus": "SUCCEEDED",
            "lastDeploymentAttemptedAt": "2020-11-28T22:40:50.512000+00:00",
            "lastDeploymentSucceededAt": "2020-11-28T22:40:50.512000+00:00",
            "spec": "proton: ServiceSpec\npipeline:\n
 my_sample_pipeline_required_input: hello\n  my_sample_pipeline_optional_input:
 bye\ninstances:\n- name: instance-svc-simple\n  environment: my-simple-
env\n  spec:\n    my_sample_service_instance_required_input: hi\n
 my_sample_service_instance_optional_input: ho\n",
            "templateMajorVersion": "1",
            "templateMinorVersion": "1",
            "templateName": "svc-simple"
        },
        "repositoryConnectionArn": "arn:aws:codestar-connections:region-
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
        "repositoryId": "myorg/myapp",
        "spec": "proton: ServiceSpec\npipeline:\n
 my_sample_pipeline_required_input: hello\n  my_sample_pipeline_optional_input:
 bye\ninstances:\n- name: instance-svc-simple\n  environment: my-simple-
env\n  spec:\n    my_sample_service_instance_required_input: hi\n
 my_sample_service_instance_optional_input: ho\n",
        "status": "ACTIVE",
        "templateName": "svc-simple"
    }
}
```


Lihat detail layanan tanpa pipeline layanan seperti yang ditunjukkan pada perintah dan respons contoh CLI berikut.

Perintah:

```
$ aws proton get-service \
    --name "simple-svc-no-pipeline"
```

Respons:

```
{
    "service": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc-without-
pipeline",
        "createdAt": "2020-11-28T22:40:50.512000+00:00",
        "lastModifiedAt": "2020-11-28T22:44:51.207000+00:00",
        "name": "simple-svc-without-pipeline",
        "spec": "proton: ServiceSpec\ninstances:\n- name: instance-svc-simple\n
 environment: my-simple-env\n  spec:\n    my_sample_service_instance_required_input:
 hi\n    my_sample_service_instance_optional_input: ho\n",
        "status": "ACTIVE",
        "templateName": "svc-simple-no-pipeline"
    }
}
```


Mengedit layanan

Anda dapat melakukan pengeditan berikut ke AWS Proton layanan.

- Edit deskripsi layanan.
- Edit layanan dengan menambahkan dan menghapus instance layanan.

Edit deskripsi layanan

Anda dapat menggunakan konsol atau AWS CLI untuk mengedit deskripsi layanan.

Konsol Manajemen AWS

Edit layanan menggunakan konsol seperti yang dijelaskan dalam langkah-langkah berikut.

Dalam daftar layanan.

1. Di AWS Proton konsol, pilih Layanan.
2. Dalam daftar layanan, pilih tombol radio di sebelah kiri layanan yang ingin Anda perbarui.
3. Pilih Edit.
4. Di halaman Konfigurasi layanan, isi formulir dan pilih Berikutnya.
5. Di halaman Konfigurasi pengaturan kustom, pilih Berikutnya.
6. Tinjau hasil edit Anda dan pilih Simpan perubahan.

Di halaman detail layanan.

1. Di AWS Proton konsol, pilih Layanan.
2. Dalam daftar layanan, pilih nama layanan yang ingin Anda edit.
3. Di halaman detail layanan, pilih Edit.
4. Di halaman Konfigurasi layanan, isi formulir dan pilih Berikutnya.
5. Di halaman Konfigurasi pengaturan khusus, isi formulir dan pilih Berikutnya.
6. Tinjau hasil edit Anda dan pilih Simpan perubahan.

AWS CLI

Mengedit deskripsi seperti yang ditunjukkan dalam contoh CLI berikut perintah dan respon.

Perintah:

```
$ aws proton update-service \
    --name "MySimpleService" \
    --description "Edit by updating description"
```

Respons:

```
{
    "service": {
        "arn": "arn:aws:proton:region-id:123456789012:service/MySimpleService",
        "branchName": "main",
        "createdAt": "2021-03-12T22:39:42.318000+00:00",
        "description": "Edit by updating description",
        "lastModifiedAt": "2021-03-12T22:44:21.975000+00:00",
        "name": "MySimpleService",
        "repositoryConnectionArn": "arn:aws:codestar-connections:region-
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "repositoryId": "my-repository/myorg-myapp",
        "status": "ACTIVE",
        "templateName": "fargate-service"
    }
}
```

Mengedit layanan untuk menambah atau menghapus instance layanan

Untuk AWS Proton layanan, Anda dapat menambahkan atau menghapus instance layanan dengan mengirimkan spesifikasi yang telah diedit. Ketentuan berikut harus dipenuhi untuk permintaan yang berhasil:

- Layanan dan pipeline Anda belum diedit atau dihapus saat Anda mengirimkan permintaan edit.
- Spesifikasi Anda yang telah diedit tidak menyertakan pengeditan yang mengubah pipeline layanan atau pengeditan ke instance layanan yang ada yang tidak akan dihapus.
- Spesifikasi Anda yang telah diedit tidak menghapus instance layanan yang ada yang memiliki komponen terlampir. Untuk menghapus instance layanan seperti itu, Anda harus memperbarui komponen terlebih dahulu untuk melepaskannya dari instance layanannya. Untuk informasi selengkapnya tentang komponen, lihatKomponen-komponen.

Instans yang gagal penghapusan adalah instance layanan di negara bagian. `DELETE_FAILED` Saat Anda meminta pengeditan layanan, AWS Proton upaya menghapus instans yang gagal dihapus untuk Anda, sebagai bagian dari proses pengeditan. Jika salah satu instance layanan Anda gagal dihapus, mungkin masih ada sumber daya yang terkait dengan instans, meskipun tidak terlihat dari konsol atau. AWS CLI Periksa sumber daya infrastruktur instans yang gagal dihapus dan bersihkan sehingga AWS Proton dapat menghapusnya untuk Anda.

Untuk kuota instans layanan untuk layanan, lihat. AWS Proton kuota Anda juga harus menyimpan setidaknya 1 instance layanan untuk layanan Anda setelah dibuat. Selama proses pembaruan, AWS Proton hitung instance layanan yang ada dan instance yang akan ditambahkan atau dihapus. Instans yang gagal penghapusan disertakan dalam hitungan ini dan Anda harus memperhitungkannya saat mengedit. `spec`

Gunakan konsol atau AWS CLI untuk menambah atau menghapus instance layanan

Konsol Manajemen AWS

Edit layanan Anda untuk menambah atau menghapus instance layanan menggunakan konsol.

Di AWS Proton konsol

1. Pada panel navigasi, silakan pilih Layanan.
2. Pilih layanan yang ingin Anda edit.
3. Pilih Edit.

4. (Opsional) Pada halaman Konfigurasi layanan, edit nama atau deskripsi layanan, lalu pilih Berikutnya.
5. Pada halaman Konfigurasi pengaturan kustom, pilih Hapus untuk menghapus instance layanan dan pilih Tambahkan instance baru untuk menambahkan instance layanan dan mengisi formulir.
6. Pilih Berikutnya.
7. Tinjau pembaruan Anda dan pilih Simpan perubahan.
8. Modal meminta Anda untuk memverifikasi penghapusan instance layanan. Ikuti instruksi dan pilih Ya, hapus.
9. Pada halaman detail layanan, lihat detail status untuk layanan Anda.

AWS CLI

Tambahkan dan hapus instance layanan dengan diedit **`spec`** seperti yang ditunjukkan pada AWS CLI contoh perintah dan tanggapan berikut.

Saat Anda menggunakan CLI, Anda `spec` harus mengecualikan instance layanan untuk dihapus dan menyertakan instance layanan yang akan ditambahkan dan instance layanan yang ada yang belum Anda tandai untuk dihapus.

Daftar berikut menunjukkan contoh `spec` sebelum pengeditan dan daftar instance layanan yang digunakan oleh spesifikasi. Spesifikasi ini digunakan dalam contoh sebelumnya untuk mengedit deskripsi layanan.

Spesifikasi:

```
proton: ServiceSpec

pipeline:
  my_sample_pipeline_optional_input: "abc"
  my_sample_pipeline_required_input: "123"

instances:
  - name: "my-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_optional_input: "def"
      my_sample_service_instance_required_input: "456"
```

```
  - name: "my-other-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_required_input: "789"
```

Contoh berikut CLI `list-service-instances` perintah dan respon menunjukkan instance aktif sebelum menambahkan atau menghapus instance layanan.

Perintah:

```
$ aws proton list-service-instances \
    --service-name "MySimpleService"
```

Respons:

```
{
    "serviceInstances": [
        {
            "arn": "arn:aws:proton:region-id:123456789012:service/MySimpleService/
service-instance/my-other-instance",
            "createdAt": "2021-03-12T22:39:42.318000+00:00",
            "deploymentStatus": "SUCCEEDED",
            "environmentName": "simple-env",
            "lastDeploymentAttemptedAt": "2021-03-12T22:39:43.109000+00:00",
            "lastDeploymentSucceededAt": "2021-03-12T22:39:43.109000+00:00",
            "name": "my-other-instance",
            "serviceName": "example-svc",
            "templateMajorVersion": "1",
            "templateMinorVersion": "0",
            "templateName": "fargate-service"
        },
        {
            "arn": "arn:aws:proton:region-id:123456789012:service/MySimpleService/
service-instance/my-instance",
            "createdAt": "2021-03-12T22:39:42.318000+00:00",
            "deploymentStatus": "SUCCEEDED",
            "environmentName": "simple-env",
            "lastDeploymentAttemptedAt": "2021-03-12T22:39:43.160000+00:00",
            "lastDeploymentSucceededAt": "2021-03-12T22:39:43.160000+00:00",
            "name": "my-instance",
            "serviceName": "example-svc",
            "serviceTemplateArn": "arn:aws:proton:region-id:123456789012:service-
template/fargate-service",
```

```
            "templateMajorVersion": "1",
            "templateMinorVersion": "0",
            "templateName": "fargate-service"
        }
    ]
}
```

Daftar berikut menunjukkan contoh diedit yang `spec` digunakan untuk menghapus dan menambahkan instance. Instance yang `my-instance` ada bernama dihapus dan instance baru bernama `yet-another-instance` ditambahkan.

Spesifikasi:

```
proton: ServiceSpec

pipeline:
  my_sample_pipeline_optional_input: "abc"
  my_sample_pipeline_required_input: "123"

instances:
  - name: "my-other-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_required_input: "789"
  - name: "yet-another-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_required_input: "789"
```

Anda dapat menggunakan `"${Proton::CURRENT_VAL}"` untuk menunjukkan nilai parameter mana yang harus dipertahankan dari aslinya`spec`, jika nilainya ada di`spec`. Gunakan `get-service` untuk melihat yang asli `spec` untuk layanan, seperti yang dijelaskan dalamLihat data layanan.

Daftar berikut menunjukkan cara yang dapat Anda gunakan `"${Proton::CURRENT_VAL}"` untuk memastikan bahwa Anda `spec` tidak menyertakan perubahan nilai parameter agar instance layanan yang ada tetap ada.

Spesifikasi:

```
proton: ServiceSpec
```

```
pipeline:
  my_sample_pipeline_optional_input: "${Proton::CURRENT_VAL}"
  my_sample_pipeline_required_input: "${Proton::CURRENT_VAL}"

instances:
  - name: "my-other-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_required_input: "${Proton::CURRENT_VAL}"
  - name: "yet-another-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_required_input: "789"
```

Daftar berikutnya menunjukkan perintah CLI dan respons untuk mengedit layanan.

Perintah:

```
$ aws proton update-service
    --name "MySimpleService" \
    --description "Edit by adding and deleting a service instance" \
    --spec "file://spec.yaml"
```

Respons:

```
{
    "service": {
        "arn": "arn:aws:proton:region-id:123456789012:service/MySimpleService",
        "branchName": "main",
        "createdAt": "2021-03-12T22:39:42.318000+00:00",
        "description": "Edit by adding and deleting a service instance",
        "lastModifiedAt": "2021-03-12T22:55:48.169000+00:00",
        "name": "MySimpleService",
        "repositoryConnectionArn": "arn:aws:codestar-connections:region-
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "repositoryId": "my-repository/myorg-myapp",
        "status": "UPDATE_IN_PROGRESS",
        "templateName": "fargate-service"
    }
}
```

`list-service-instances`Perintah dan respons berikut mengonfirmasi bahwa instance yang `my-instance` ada bernama dihapus dan instance baru bernama `yet-another-instance` ditambahkan.

Perintah:

```
$ aws proton list-service-instances \
    --service-name "MySimpleService"
```

Respons:

```
{
    "serviceInstances": [
        {
            "arn": "arn:aws:proton:region-id:123456789012:service/MySimpleService/
service-instance/yet-another-instance",
            "createdAt": "2021-03-12T22:39:42.318000+00:00",
            "deploymentStatus": "SUCCEEDED",
            "environmentName": "simple-env",
            "lastDeploymentAttemptedAt": "2021-03-12T22:56:01.565000+00:00",
            "lastDeploymentSucceededAt": "2021-03-12T22:56:01.565000+00:00",
            "name": "yet-another-instance",
            "serviceName": "MySimpleService",
            "templateMajorVersion": "1",
            "templateMinorVersion": "0",
            "templateName": "fargate-service"
        },
        {
            "arn": "arn:aws:proton:region-id:123456789012:service/MySimpleService/
service-instance/my-other-instance",
            "createdAt": "2021-03-12T22:39:42.318000+00:00",
            "deploymentStatus": "SUCCEEDED",
            "environmentName": "simple-env",
            "lastDeploymentAttemptedAt": "2021-03-12T22:39:43.109000+00:00",
            "lastDeploymentSucceededAt": "2021-03-12T22:39:43.109000+00:00",
            "name": "my-other-instance",
            "serviceName": "MySimpleService",
            "templateMajorVersion": "1",
            "templateMinorVersion": "0",
            "templateName": "fargate-service"
        }
    ]
```

```
}
```

Apa yang terjadi ketika Anda menambahkan atau menghapus instance layanan

Setelah Anda mengirimkan suntingan layanan untuk menghapus dan menambahkan instance layanan, AWS Proton lakukan tindakan berikut.

- Menetapkan layanan ke`UPDATE_IN_PROGRESS`.
- Jika layanan memiliki pipeline, tetapkan statusnya ke `IN_PROGRESS` dan blokir tindakan pipeline.
- Menetapkan instance layanan apa pun yang akan dihapus. `DELETE_IN_PROGRESS`
- Memblokir tindakan layanan.
- Memblokir tindakan pada instance layanan yang ditandai untuk dihapus.
- Membuat instance layanan baru.
- Menghapus instance yang Anda cantumkan untuk dihapus.
- Mencoba untuk menghapus instance deletion-failed.
- Setelah penambahan dan penghapusan selesai, berikan kembali jalur pipa layanan (jika ada), atur layanan Anda `ACTIVE` dan aktifkan tindakan layanan dan saluran pipa.

AWS Proton mencoba untuk menengahi kembali mode kegagalan sebagai berikut.

- Jika satu atau beberapa instance layanan gagal dibuat, AWS Proton coba hapus penyediaan semua instance layanan yang baru dibuat dan kembalikan `spec` ke status sebelumnya. Itu tidak menghapus instance layanan apa pun dan tidak mengubah pipeline dengan cara apa pun.
- Jika satu atau beberapa instance layanan gagal dihapus, berikan AWS Proton kembali pipeline tanpa instans yang dihapus. `spec`Ini diperbarui untuk menyertakan instance yang ditambahkan dan untuk mengecualikan instance yang ditandai untuk dihapus.
- Jika pipeline gagal dalam penyediaan, rollback tidak dicoba dan layanan dan pipeline mencerminkan status pembaruan yang gagal.

Penandaan dan pengeditan layanan

Saat Anda menambahkan instance layanan sebagai bagian dari pengeditan layanan, tag AWS terkelola menyebar ke dan secara otomatis dibuat untuk instance baru dan sumber daya yang disediakan. Jika Anda membuat tag baru, tag tersebut hanya diterapkan ke instance baru.

Tag terkelola pelanggan layanan yang ada juga menyebar ke instance baru. Untuk informasi selengkapnya, lihat AWS Proton sumber daya dan penandaan.

Hapus layanan

Anda dapat menghapus AWS Proton layanan, dengan instance dan pipeline, dengan menggunakan AWS Proton konsol atau file. AWS CLI

Anda tidak dapat menghapus layanan yang memiliki instance layanan apa pun dengan komponen terlampir. Untuk menghapus layanan seperti itu, Anda harus terlebih dahulu memperbarui semua komponen terlampir untuk melepaskannya dari instance layanannya. Untuk informasi selengkapnya tentang komponen, lihatKomponen-komponen.

Konsol Manajemen AWS

Hapus layanan menggunakan konsol seperti yang dijelaskan dalam langkah-langkah berikut.

Di halaman detail layanan.

1. Di AWS Proton konsol, pilih Layanan.
2. Dalam daftar layanan, pilih nama layanan yang ingin Anda hapus.
3. Pada halaman detail layanan, pilih Tindakan dan kemudian Hapus.
4. Modal meminta Anda untuk mengonfirmasi tindakan hapus.
5. Ikuti instruksi dan pilih Ya, hapus.

AWS CLI

Hapus layanan seperti yang ditunjukkan pada contoh perintah dan respons CLI berikut.

Perintah:

```
$ aws proton delete-service \
    --name "simple-svc"
```

Respons:

```
{
    "service": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc",
```

```
        "branchName": "mainline",
        "createdAt": "2020-11-28T22:40:50.512000+00:00",
        "description": "Edit by updating description",
        "lastModifiedAt": "2020-11-29T00:30:39.248000+00:00",
        "name": "simple-svc",
        "repositoryConnectionArn": "arn:aws:codestar-connections:region-
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "repositoryId": "myorg/myapp",
        "status": "DELETE_IN_PROGRESS",
        "templateName": "fargate-service"
    }
}
```

Lihat data contoh layanan

Pelajari cara melihat data detail instance AWS Proton layanan. Anda dapat menggunakan konsol atau AWS CLI.

Contoh layanan milik layanan. Anda hanya dapat membuat atau menghapus instance dalam konteks layanan mengedit, membuat, dan menghapus tindakan. Untuk mempelajari cara menambah dan menghapus instance dari layanan, lihatMengedit layanan.

Konsol Manajemen AWS

Daftar dan lihat detail instance layanan menggunakan AWS Proton konsol seperti yang ditunjukkan pada langkah-langkah berikut.

1. Untuk melihat daftar instance layanan Anda, pilih Instans layanan di panel navigasi.
2. Untuk melihat data detail, pilih nama instance layanan.

   Lihat data detail instance layanan Anda.

AWS CLI

Daftar dan lihat detail contoh layanan seperti yang ditunjukkan dalam perintah dan tanggapan contoh CLI berikut.

Perintah:

```
$ aws proton list-service-instances
```

Respons:

```
{
    "serviceInstances": [
        {
            "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/
service-instance/instance-one",
            "createdAt": "2020-11-28T22:40:50.512000+00:00",
            "deploymentStatus": "SUCCEEDED",
            "environmentArn": "arn:aws:proton:region-id:123456789012:environment/
simple-env",
            "lastDeploymentAttemptedAt": "2020-11-28T22:40:50.512000+00:00",
            "lastDeploymentSucceededAt": "2020-11-28T22:40:50.512000+00:00",
            "name": "instance-one",
            "serviceName": "simple-svc",
            "templateMajorVersion": "1",
            "templateMinorVersion": "0",
            "templateName": "fargate-service"
        }
    ]
}
```

Perintah:

```
$ aws proton get-service-instance \
    --name "instance-one" \
    --service-name "simple-svc"
```

Respons:

```
{
    "serviceInstance": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/service-
instance/instance-one",
        "createdAt": "2020-11-28T22:40:50.512000+00:00",
        "deploymentStatus": "SUCCEEDED",
        "environmentName": "simple-env",
        "lastDeploymentAttemptedAt": "2020-11-28T22:40:50.512000+00:00",
        "lastDeploymentSucceededAt": "2020-11-28T22:40:50.512000+00:00",
        "name": "instance-one",
        "serviceName": "simple-svc",
        "spec": "proton: ServiceSpec\npipeline:\n
 my_sample_pipeline_optional_input: hello world\n
```

```
 my_sample_pipeline_required_input: pipeline up\ninstances:\n- name: instance-one\n
 environment: my-simple-env\n  spec:\n    my_sample_service_instance_optional_input:
 Ola\n    my_sample_service_instance_required_input: Ciao\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "0",
        "templateName": "svc-simple"
    }
}
```

Memperbarui instance layanan

Pelajari cara memperbarui instance AWS Proton layanan dan membatalkan pembaruan.

Contoh layanan milik layanan. Anda hanya dapat membuat atau menghapus instance dalam konteks layanan mengedit, membuat, dan menghapus tindakan. Untuk mempelajari cara menambah dan menghapus instance dari layanan, lihatMengedit layanan.

Ada empat mode untuk memperbarui instance layanan seperti yang dijelaskan dalam daftar berikut. Saat menggunakan AWS CLI, `deployment-type` bidang mendefinisikan mode. Saat menggunakan konsol, mode ini dipetakan ke Edit dan Perbarui ke versi minor terbaru dan Perbarui ke tindakan versi utama terbaru yang diturunkan dari Tindakan di halaman detail instance layanan.

`NONE`

Dalam mode ini, penerapan tidak terjadi. Hanya parameter metadata yang diminta yang diperbarui.

`CURRENT_VERSION`

Dalam mode ini, instance layanan disebarkan dan diperbarui dengan spesifikasi baru yang Anda berikan. Hanya parameter yang diminta yang diperbarui. Jangan sertakan parameter versi minor atau mayor saat Anda menggunakan ini`deployment-type`.

`MINOR_VERSION`

Dalam mode ini, instance layanan disebarkan dan diperbarui dengan versi minor yang dipublikasikan dan direkomendasikan (terbaru) dari versi utama saat ini yang digunakan secara default. Anda juga dapat menentukan versi minor yang berbeda dari versi utama saat ini yang digunakan.

`MAJOR_VERSION`

Dalam mode ini, instance layanan disebarkan dan diperbarui dengan versi mayor dan minor yang dipublikasikan, direkomendasikan (terbaru) dari template saat ini secara default. Anda juga dapat menentukan versi mayor yang berbeda yang lebih tinggi dari versi utama yang digunakan dan versi minor (opsional).

Anda dapat mencoba membatalkan penerapan pembaruan instans layanan jika `deploymentStatus` ada`IN_PROGRESS`. AWS Proton upaya untuk membatalkan penyebaran. Pembatalan yang berhasil tidak dijamin.

Saat Anda membatalkan penerapan pembaruan, AWS Proton upaya untuk membatalkan penerapan seperti yang tercantum dalam langkah-langkah berikut.

- Menetapkan status penerapan ke`CANCELLING`.
- Menghentikan penerapan dalam proses dan menghapus sumber daya baru apa pun yang dibuat oleh penerapan saat. `IN_PROGRESS`
- Menetapkan status penerapan ke`CANCELLED`.
- Mengembalikan status sumber daya ke keadaan sebelum penerapan dimulai.

Untuk informasi selengkapnya tentang membatalkan penerapan instance layanan, lihat CancelServiceInstanceDeploymentdi Referensi AWS Proton API.

Gunakan konsol atau AWS CLI untuk membuat pembaruan atau membatalkan penerapan pembaruan.

Konsol Manajemen AWS

Perbarui instance layanan menggunakan konsol dengan mengikuti langkah-langkah ini.

1. Di AWS Proton konsol, pilih Instans layanan di panel navigasi.
2. Dalam daftar instance layanan, pilih nama instance layanan yang ingin Anda perbarui.
3. Pilih Tindakan dan kemudian pilih salah satu opsi pembaruan, Edit untuk memperbarui spesifikasi atau Tindakan dan kemudian Perbarui ke versi minor terbaru, atau Perbarui ke versi utama terbaru.
4. Isi setiap formulir dan pilih Berikutnya hingga Anda mencapai halaman Ulasan.

5. Tinjau hasil edit Anda dan pilih Perbarui.

AWS CLI

Perbarui instance layanan ke versi minor baru seperti yang ditunjukkan dalam perintah dan tanggapan contoh CLI.

Saat memperbarui instance layanan dengan modifikasi`spec`, Anda dapat menggunakan `"${Proton::CURRENT_VAL}"` untuk menunjukkan nilai parameter mana yang akan dipertahankan dari aslinya`spec`, jika nilainya ada di`spec`. Gunakan `get-service` untuk melihat asli `spec` untuk instance layanan, seperti yang dijelaskan dalamLihat data layanan.

Contoh berikut menunjukkan bagaimana Anda dapat menggunakan `"${Proton::CURRENT_VAL}"` dalam`spec`.

Spesifikasi:

```
proton: ServiceSpec

pipeline:
  my_sample_pipeline_optional_input: "${Proton::CURRENT_VAL}"
  my_sample_pipeline_required_input: "${Proton::CURRENT_VAL}"

instances:
  - name: "my-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_optional_input: "${Proton::CURRENT_VAL}"
      my_sample_service_instance_required_input: "${Proton::CURRENT_VAL}"
  - name: "my-other-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_required_input: "789"
```

Perintah: untuk memperbarui

```
$ aws proton update-service-instance \
    --name "instance-one" \
    --service-name "simple-svc" \
    --spec "file://service-spec.yaml" \
    --template-major-version "1" \
```

```
    --template-minor-version "1" \
    --deployment-type "MINOR_VERSION"
```

Respons:

```
{
    "serviceInstance": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/service-
instance/instance-one",
        "createdAt": "2021-04-02T21:29:59.962000+00:00",
        "deploymentStatus": "IN_PROGRESS",
        "environmentName": "arn:aws:proton:region-id:123456789012:environment/
simple-env",
        "lastDeploymentAttemptedAt": "2021-04-02T21:38:00.823000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T21:29:59.962000+00:00",
        "name": "instance-one",
        "serviceName": "simple-svc",
        "templateMajorVersion": "1",
        "templateMinorVersion": "0",
        "templateName": "svc-simple"
    }
}
```

Perintah: untuk mendapatkan dan mengkonfirmasi status

```
$ aws proton get-service-instance \
    --name "instance-one" \
    --service-name "simple-svc"
```

Respons:

```
{
    "serviceInstance": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/service-
instance/instance-one",
        "createdAt": "2021-04-02T21:29:59.962000+00:00",
        "deploymentStatus": "SUCCEEDED",
        "environmentName": "simple-env",
        "lastDeploymentAttemptedAt": "2021-04-02T21:38:00.823000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T21:38:00.823000+00:00",
        "name": "instance-one",
        "serviceName": "simple-svc",
```

```
        "spec": "proton: ServiceSpec\n\npipeline:\n
 my_sample_pipeline_optional_input: \"abc\"\n  my_sample_pipeline_required_input:
 \"123\"\n\ninstances:\n  - name: \"instance-one\"\n    environment: \"simple-
env\"\n    spec:\n      my_sample_service_instance_optional_input: \"def\"\n
      my_sample_service_instance_required_input: \"456\"\n  - name: \"my-
other-instance\"\n    environment: \"kls-simple-env\"\n    spec:\n
 my_sample_service_instance_required_input: \"789\"\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
        "templateName": "svc-simple"
    }
}
```

Konsol Manajemen AWS

Batalkan penyebaran instance layanan menggunakan konsol seperti yang ditunjukkan pada langkah-langkah berikut.

1. Di AWS Proton konsol, pilih Instans layanan di panel navigasi.
2. Dalam daftar instance layanan, pilih nama instance layanan dengan pembaruan penyebaran yang ingin Anda batalkan.
3. Jika status penyebaran pembaruan Anda sedang berlangsung, di halaman detail instance layanan, pilih Tindakan, lalu Batalkan penerapan.
4. Modal meminta Anda untuk mengkonfirmasi pembatalan. Pilih Batalkan penerapan.
5. Status penyebaran pembaruan Anda diatur ke Membatalkan dan kemudian Dibatalkan untuk menyelesaikan pembatalan.

AWS CLI

Batalkan pembaruan penyebaran instance layanan IN_PROGRESS ke versi minor baru 2 seperti yang ditunjukkan pada perintah dan tanggapan contoh CLI berikut.

Kondisi tunggu disertakan dalam template yang digunakan untuk contoh ini sehingga pembatalan dimulai sebelum penerapan pembaruan berhasil.

Perintah: untuk membatalkan

```
$ aws proton cancel-service-instance-deployment \
    --service-instance-name "instance-one" \
```

```
    --service-name "simple-svc"
```

Respons:

```
{
    "serviceInstance": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/service-
instance/instance-one",
        "createdAt": "2021-04-02T21:29:59.962000+00:00",
        "deploymentStatus": "CANCELLING",
        "environmentName": "simple-env",
        "lastDeploymentAttemptedAt": "2021-04-02T21:45:15.406000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T21:38:00.823000+00:00",
        "name": "instance-one",
        "serviceName": "simple-svc",
        "spec": "proton: ServiceSpec\npipeline:\n
 my_sample_pipeline_optional_input: abc\n  my_sample_pipeline_required_input:
 '123'\ninstances:\n- name: my-instance\n  environment: MySimpleEnv
\n  spec:\n    my_sample_service_instance_optional_input: def\n
 my_sample_service_instance_required_input: '456'\n- name: my-other-instance\n
 environment: MySimpleEnv\n  spec:\n    my_sample_service_instance_required_input:
 '789'\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
        "templateName": "svc-simple"
    }
}
```

Perintah: untuk mendapatkan dan mengkonfirmasi status

```
$ aws proton get-service-instance \
    --name "instance-one" \
    --service-name "simple-svc"
```

Respons:

```
{
    "serviceInstance": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/service-
instance/instance-one",
        "createdAt": "2021-04-02T21:29:59.962000+00:00",
        "deploymentStatus": "CANCELLED",
```

```
        "deploymentStatusMessage": "User initiated cancellation.",
        "environmentName": "simple-env",
        "lastDeploymentAttemptedAt": "2021-04-02T21:45:15.406000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T21:38:00.823000+00:00",
        "name": "instance-one",
        "serviceName": "simple-svc",
        "spec": "proton: ServiceSpec\n\npipeline:\n
 my_sample_pipeline_optional_input: \"abc\"\n  my_sample_pipeline_required_input:
 \"123\"\n\ninstances:\n  - name: \"instance-one\"\n    environment: \"simple-
env\"\n    spec:\n      my_sample_service_instance_optional_input: \"def\"\n
      my_sample_service_instance_required_input: \"456\"\n  - name: \"my-
other-instance\"\n    environment: \"kls-simple-env\"\n    spec:\n
 my_sample_service_instance_required_input: \"789\"\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
        "templateName": "svc-simple"
    }
}
```

Memperbarui saluran layanan

Pelajari cara memperbarui saluran AWS Proton layanan dan membatalkan pembaruan.

Pipa layanan milik layanan. Anda hanya dapat membuat atau menghapus pipeline dalam konteks tindakan membuat dan menghapus layanan.

Ada empat mode untuk memperbarui pipa layanan seperti yang dijelaskan dalam daftar berikut. Saat menggunakan AWS CLI, `deployment-type` bidang mendefinisikan mode. Saat Anda menggunakan konsol, mode ini dipetakan ke pipeline Edit dan Perbarui ke versi yang direkomendasikan.

`NONE`

Dalam mode ini, penerapan tidak terjadi. Hanya parameter metadata yang diminta yang diperbarui.

`CURRENT_VERSION`

Dalam mode ini, pipeline layanan dikerahkan dan diperbarui dengan spesifikasi baru yang Anda berikan. Hanya parameter yang diminta yang diperbarui. Jangan sertakan parameter versi minor atau mayor saat Anda menggunakan ini`deployment-type`.

`MINOR_VERSION`

Dalam mode ini, pipeline layanan disebarkan dan diperbarui dengan versi minor yang dipublikasikan dan direkomendasikan (terbaru) dari versi utama saat ini yang digunakan secara default. Anda juga dapat menentukan versi minor yang berbeda dari versi utama saat ini yang digunakan.

`MAJOR_VERSION`

Dalam mode ini, pipeline layanan dikerahkan dan diperbarui dengan versi mayor dan minor yang dipublikasikan, direkomendasikan (terbaru) dari template saat ini secara default. Anda juga dapat menentukan versi mayor yang berbeda yang lebih tinggi dari versi utama yang digunakan dan versi minor (opsional).

Anda dapat mencoba membatalkan penerapan pembaruan saluran pipa layanan jika `deploymentStatus` ada`IN_PROGRESS`. AWS Proton upaya untuk membatalkan penyebaran. Pembatalan yang berhasil tidak dijamin.

Saat Anda membatalkan penerapan pembaruan, AWS Proton upaya untuk membatalkan penerapan seperti yang tercantum dalam langkah-langkah berikut.

- Menetapkan status penerapan ke`CANCELLING`.
- Menghentikan penerapan dalam proses dan menghapus sumber daya baru apa pun yang dibuat oleh penerapan saat. `IN_PROGRESS`
- Menetapkan status penerapan ke`CANCELLED`.
- Mengembalikan status sumber daya ke keadaan sebelum penerapan dimulai.

Untuk informasi selengkapnya tentang membatalkan penerapan pipeline layanan, lihat CancelServicePipelineDeploymentdi Referensi AWS Proton API.

Gunakan konsol atau AWS CLI untuk membuat pembaruan atau membatalkan penerapan pembaruan.

Konsol Manajemen AWS

Perbarui saluran layanan menggunakan konsol seperti yang dijelaskan dalam langkah-langkah berikut.

1. Di AWS Proton konsol, pilih Layanan.
2. Dalam daftar layanan, pilih nama layanan yang ingin Anda perbarui pipeline.
3. Ada dua tab pada halaman detail layanan, Ikhtisar dan Pipeline. Pilih Pipeline.
4. Jika Anda ingin memperbarui spesifikasi, pilih Edit Pipeline dan isi setiap formulir dan pilih Berikutnya hingga Anda menyelesaikan formulir akhir dan kemudian pilih Perbarui pipeline.

   Jika Anda ingin memperbarui ke versi baru dan ada ikon informasi yang menunjukkan versi baru tersedia di template Pipeline, pilih nama versi template baru.

   a. Pilih Perbarui ke versi yang direkomendasikan.
   b. Isi setiap formulir dan pilih Berikutnya sampai Anda menyelesaikan formulir akhir dan memilih Perbarui.

AWS CLI

Perbarui pipeline layanan ke versi minor baru seperti yang ditunjukkan pada perintah dan tanggapan contoh CLI berikut.

Saat memperbarui pipeline layanan dengan modifikasi`spec`, Anda dapat menggunakan `"${Proton::CURRENT_VAL}"` untuk menunjukkan nilai parameter mana yang akan dipertahankan dari aslinya`spec`, jika nilainya ada di`spec`. Gunakan `get-service` untuk melihat aslinya `spec` untuk pipeline layanan, seperti yang dijelaskan dalamLihat data layanan.

Contoh berikut menunjukkan bagaimana Anda dapat menggunakan `"${Proton::CURRENT_VAL}"` dalam`spec`.

Spesifikasi:

```
proton: ServiceSpec

pipeline:
  my_sample_pipeline_optional_input: "${Proton::CURRENT_VAL}"
  my_sample_pipeline_required_input: "${Proton::CURRENT_VAL}"

instances:
  - name: "my-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_optional_input: "${Proton::CURRENT_VAL}"
      my_sample_service_instance_required_input: "${Proton::CURRENT_VAL}"
```

```
  - name: "my-other-instance"
    environment: "simple-env"
    spec:
      my_sample_service_instance_required_input: "789"
```

Perintah: untuk memperbarui

```
$ aws proton update-service-pipeline \
    --service-name "simple-svc" \
    --spec "file://service-spec.yaml" \
    --template-major-version "1" \
    --template-minor-version "1" \
    --deployment-type "MINOR_VERSION"
```

Respons:

```
{
    "pipeline": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/pipeline/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "createdAt": "2021-04-02T21:29:59.962000+00:00",
        "deploymentStatus": "IN_PROGRESS",
        "lastDeploymentAttemptedAt": "2021-04-02T21:39:28.991000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T21:29:59.962000+00:00",
        "spec": "proton: ServiceSpec\n\npipeline:\n
 my_sample_pipeline_optional_input: \"abc\"\n  my_sample_pipeline_required_input:
 \"123\"\n\ninstances:\n  - name: \"my-instance\"\n    environment: \"MySimpleEnv
\"\n    spec:\n      my_sample_service_instance_optional_input: \"def
\"\n      my_sample_service_instance_required_input: \"456\"\n  - name:
 \"my-other-instance\"\n    environment: \"MySimpleEnv\"\n    spec:\n
 my_sample_service_instance_required_input: \"789\"\n",
        "templateMajorVersion": "1",
        "templateMinorVersion": "0",
        "templateName": "svc-simple"
    }
}
```

Perintah: untuk mendapatkan dan mengkonfirmasi status

```
$ aws proton get-service \
    --name "simple-svc"
```

Respons:


```
{
    "service": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc",
        "branchName": "main",
        "createdAt": "2021-04-02T21:29:59.962000+00:00",
        "lastModifiedAt": "2021-04-02T21:30:54.364000+00:00",
        "name": "simple-svc",
        "pipeline": {
            "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/
pipeline",
            "createdAt": "2021-04-02T21:29:59.962000+00:00",
            "deploymentStatus": "SUCCEEDED",
            "lastDeploymentAttemptedAt": "2021-04-02T21:39:28.991000+00:00",
            "lastDeploymentSucceededAt": "2021-04-02T21:39:28.991000+00:00",
            "spec": "proton: ServiceSpec\n\npipeline:\n
 my_sample_pipeline_optional_input: \"abc\"\n  my_sample_pipeline_required_input:
 \"123\"\n\ninstances:\n  - name: \"instance-one\"\n    environment: \"simple-
env\"\n    spec:\n      my_sample_service_instance_optional_input: \"def
\"\n      my_sample_service_instance_required_input: \"456\"\n  - name:
 \"my-other-instance\"\n    environment: \"simple-env\"\n    spec:\n
 my_sample_service_instance_required_input: \"789\"\n",
            "templateMajorVersion": "1",
            "templateMinorVersion": "1",
            "templateName": "svc-simple"
        },
        "repositoryConnectionArn": "arn:aws:codestar-connections:region-
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "repositoryId": "repo-name/myorg-myapp",
        "spec": "proton: ServiceSpec\n\npipeline:\n
 my_sample_pipeline_optional_input: \"abc\"\n  my_sample_pipeline_required_input:
 \"123\"\n\ninstances:\n  - name: \"instance-one\"\n    environment: \"simple-
env\"\n    spec:\n      my_sample_service_instance_optional_input: \"def
\"\n      my_sample_service_instance_required_input: \"456\"\n  - name:
 \"my-other-instance\"\n    environment: \"simple-env\"\n    spec:\n
 my_sample_service_instance_required_input: \"789\"\n",
        "status": "ACTIVE",
        "templateName": "svc-simple"
    }
}
```

Konsol Manajemen AWS

Batalkan penerapan pipeline layanan menggunakan konsol seperti yang ditunjukkan pada langkah-langkah berikut.

1. Di AWS Proton konsol, pilih Layanan di panel navigasi.
2. Dalam daftar layanan, pilih nama layanan yang memiliki pipeline dengan pembaruan penyebaran yang ingin Anda batalkan.
3. Di halaman detail layanan, pilih tab Pipeline.
4. Jika status penerapan pembaruan Anda sedang berlangsung, di halaman detail pipeline layanan, pilih Batalkan penerapan.
5. Modal meminta Anda untuk mengkonfirmasi pembatalan. Pilih Batalkan penerapan.
6. Status penyebaran pembaruan Anda diatur ke Membatalkan dan kemudian Dibatalkan untuk menyelesaikan pembatalan.

AWS CLI

Batalkan pembaruan penyebaran pipeline layanan IN_PROGRESS ke versi minor 2 seperti yang ditunjukkan pada perintah dan tanggapan contoh CLI berikut.

Kondisi tunggu disertakan dalam template yang digunakan untuk contoh ini sehingga pembatalan dimulai sebelum penerapan pembaruan berhasil.

Perintah: untuk membatalkan

```
$ aws proton cancel-service-pipeline-deployment \
    --service-name "simple-svc"
```

Respons:

```
{
    "pipeline": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/pipeline",
        "createdAt": "2021-04-02T21:29:59.962000+00:00",
        "deploymentStatus": "CANCELLING",
        "lastDeploymentAttemptedAt": "2021-04-02T22:02:45.095000+00:00",
        "lastDeploymentSucceededAt": "2021-04-02T21:39:28.991000+00:00",
        "templateMajorVersion": "1",
        "templateMinorVersion": "1",
```

```
        "templateName": "svc-simple"
    }
}
```

Perintah: untuk mendapatkan dan mengkonfirmasi status

```
$ aws proton get-service \
    --name "simple-svc"
```

Respons:

```
{
    "service": {
        "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc",
        "branchName": "main",
        "createdAt": "2021-04-02T21:29:59.962000+00:00",
        "lastModifiedAt": "2021-04-02T21:30:54.364000+00:00",
        "name": "simple-svc",
        "pipeline": {
            "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/
pipeline",
            "createdAt": "2021-04-02T21:29:59.962000+00:00",
            "deploymentStatus": "CANCELLED",
            "deploymentStatusMessage": "User initiated cancellation.",
            "lastDeploymentAttemptedAt": "2021-04-02T22:02:45.095000+00:00",
            "lastDeploymentSucceededAt": "2021-04-02T21:39:28.991000+00:00",
            "spec": "proton: ServiceSpec\n\npipeline:\n
 my_sample_pipeline_optional_input: \"abc\"\n  my_sample_pipeline_required_input:
 \"123\"\n\ninstances:\n  - name: \"instance-one\"\n    environment: \"simple-
env\"\n    spec:\n      my_sample_service_instance_optional_input: \"def
\"\n      my_sample_service_instance_required_input: \"456\"\n  - name:
 \"my-other-instance\"\n    environment: \"simple-env\"\n    spec:\n
 my_sample_service_instance_required_input: \"789\"\n",
            "templateMajorVersion": "1",
            "templateMinorVersion": "1",
            "templateName": "svc-simple"
        },
        "repositoryConnectionArn": "arn:aws:codestar-connections:region-
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "repositoryId": "repo-name/myorg-myapp",
        "spec": "proton: ServiceSpec\n\npipeline:\n
 my_sample_pipeline_optional_input: \"abc\"\n  my_sample_pipeline_required_input:
 \"123\"\n\ninstances:\n  - name: \"instance-one\"\n    environment: \"simple-
```

```
env\"\n    spec:\n      my_sample_service_instance_optional_input: \"def
\"\n      my_sample_service_instance_required_input: \"456\"\n  - name:
 \"my-other-instance\"\n    environment: \"simple-env\"\n    spec:\n
 my_sample_service_instance_required_input: \"789\"\n",
        "status": "ACTIVE",
        "templateName": "svc-simple"
    }
}
```

AWS Proton komponen

Komponen adalah jenis sumber AWS Proton daya. Mereka menambahkan fleksibilitas ke template layanan. Komponen menyediakan tim platform dengan mekanisme untuk memperluas pola infrastruktur inti, dan menentukan perlindungan yang memberdayakan pengembang untuk mengelola aspek infrastruktur aplikasi mereka.

Dalam AWS Proton administrator mendefinisikan infrastruktur standar yang digunakan di seluruh tim pengembangan dan aplikasi. Namun, tim pengembangan mungkin perlu menyertakan sumber daya tambahan untuk kasus penggunaan spesifik mereka, seperti antrian Amazon Simple Queue Service (Amazon SQS) atau tabel Amazon DynamoDB. Sumber daya khusus aplikasi ini mungkin sering berubah, terutama selama pengembangan aplikasi awal. Mempertahankan perubahan yang sering terjadi pada templat yang ditulis administrator ini mungkin sulit untuk dikelola dan diskalakan — administrator perlu mempertahankan lebih banyak templat tanpa nilai tambah administrator yang sebenarnya. Alternatif—membiarkan pengembang aplikasi membuat template untuk aplikasi mereka —juga tidak ideal, karena menghilangkan kemampuan administrator untuk membakukan komponen arsitektur utama, seperti tugas. AWS Fargate Di sinilah komponen masuk.

Dengan komponen, pengembang dapat menambahkan sumber daya tambahan ke aplikasi mereka, di atas dan di luar apa yang didefinisikan administrator dalam template lingkungan dan layanan. Pengembang kemudian melampirkan komponen ke instance layanan. AWS Proton menyediakan sumber daya infrastruktur yang ditentukan oleh komponen seperti halnya menyediakan sumber daya untuk lingkungan dan instance layanan.

Komponen dapat membaca input instance layanan dan memberikan output ke instance layanan, untuk pengalaman yang terintegrasi penuh. Misalnya, jika komponen menambahkan bucket Amazon Simple Storage Service (Amazon S3) Simple Storage Service (Amazon S3) untuk digunakan oleh instance layanan, template komponen dapat mempertimbangkan nama instance lingkungan dan layanan untuk penamaan bucket. Saat AWS Proton merender template layanan untuk menyediakan instance layanan, instance layanan dapat merujuk ke bucket dan menggunakannya.

Komponen yang AWS Proton saat ini mendukung adalah komponen yang didefinisikan secara langsung. Anda meneruskan file Infrastructure as Code (IAc) yang mendefinisikan infrastruktur komponen secara langsung ke AWS Proton API atau konsol. Ini berbeda dari lingkungan atau layanan, di mana Anda mendefinisikan IAc dalam bundel template dan mendaftarkan bundel sebagai sumber daya template, kemudian menggunakan sumber daya template untuk membuat lingkungan atau layanan.

> **Note**
>
> Komponen yang didefinisikan secara langsung memungkinkan pengembang untuk menentukan infrastruktur tambahan dan menyediakannya. AWS Proton ketentuan semua komponen yang didefinisikan secara langsung berjalan di lingkungan yang sama menggunakan peran yang sama AWS Identity and Access Management (IAM).

Administrator dapat mengontrol apa yang dapat dilakukan pengembang dengan komponen dengan dua cara:

- Sumber komponen yang didukung — Administrator dapat mengizinkan lampiran komponen ke instance layanan berdasarkan properti versi templat AWS Proton layanan. Secara default, pengembang tidak dapat melampirkan komponen ke instance layanan.

  Untuk informasi selengkapnya tentang properti ini, lihat supportedComponentSourcesparameter tindakan CreateServiceTemplateVersionAPI di Referensi AWS Proton API.

  > **Note**
  >
  > Saat Anda menggunakan sinkronisasi templat, AWS Proton buat versi templat layanan secara implisit saat Anda melakukan perubahan pada bundel templat layanan di repositori. Dalam hal ini, alih-alih menentukan sumber komponen yang didukung selama pembuatan versi templat layanan, Anda menentukan properti ini dalam file yang terkait dengan setiap versi utama templat layanan. Untuk informasi selengkapnya, lihat the section called “Menyinkronkan templat layanan”.

- Peran komponen — Administrator dapat menetapkan peran komponen ke lingkungan. AWS Proton mengasumsikan peran ini ketika menyediakan infrastruktur yang ditentukan oleh komponen yang didefinisikan secara langsung di lingkungan. Oleh karena itu, peran komponen mencakup infrastruktur yang dapat ditambahkan pengembang menggunakan komponen yang didefinisikan secara langsung di lingkungan. Dengan tidak adanya peran komponen, pengembang tidak dapat membuat komponen yang didefinisikan secara langsung di lingkungan.

  Untuk informasi selengkapnya tentang menetapkan peran komponen, lihat componentRoleArnparameter tindakan CreateEnvironmentAPI di Referensi AWS Proton API.

> **Note**
>
> Peran komponen tidak digunakan di Penyediaan yang dikelola sendiri lingkungan.

Topik

- Bagaimana komponen dibandingkan dengan AWS Proton sumber daya lain?
- Komponen di AWS Proton konsol
- Komponen dalam AWS Proton API dan AWS CLI
- Pertanyaan yang sering diajukan komponen
- Status komponen
- Infrastruktur komponen sebagai file kode
- CloudFormation Contoh komponen

Bagaimana komponen dibandingkan dengan AWS Proton sumber daya lain?

Dalam banyak hal, komponen mirip dengan AWS Proton sumber daya lainnya. Infrastruktur mereka didefinisikan dalam file template IAC, ditulis dalam format CloudFormation YAMAL atau Terraform HCL. AWS Proton dapat menyediakan infrastruktur komponen menggunakan penyediaan AWS-managed atau self-managed provisioning.

Komponen, bagaimanapun, berbeda dari AWS Proton sumber daya lain dalam beberapa hal:

- Status terpisah — Komponen dirancang untuk dilampirkan ke instance layanan dan untuk memperluas infrastrukturnya, tetapi juga dapat dalam keadaan terpisah, di mana mereka tidak dilampirkan ke instance layanan apa pun. Untuk informasi selengkapnya tentang status komponen, lihatthe section called “Status komponen”.
- Tidak ada skema - Komponen tidak memiliki skema terkait seperti yang dimiliki bundel template. Input komponen didefinisikan oleh layanan. Sebuah komponen dapat mengkonsumsi input ketika dilampirkan ke instance layanan.
- Tidak ada komponen yang dikelola pelanggan — AWS Proton selalu menyediakan infrastruktur komponen untuk Anda. Tidak ada versi komponen sumber daya Anda sendiri. Untuk informasi selengkapnya tentang lingkungan yang dikelola pelanggan, lihat. the section called “Buat”

- Tidak ada sumber daya template - Komponen yang didefinisikan secara langsung tidak memiliki sumber daya template terkait yang mirip dengan templat lingkungan dan layanan. Anda menyediakan file template IAC langsung ke komponen. Demikian pula, Anda langsung menyediakan manifes yang mendefinisikan bahasa template dan mesin rendering untuk menyediakan infrastruktur komponen. Anda membuat file template dan manifes dengan cara yang mirip dengan membuat bundel template. Namun, dengan komponen yang didefinisikan secara langsung, tidak ada persyaratan untuk menyimpan file IAC sebagai bundel di lokasi tertentu, dan Anda tidak membuat sumber daya template di AWS Proton luar file IAc.
- Tidak ada penyediaan CodeBuild berbasis - Anda tidak dapat menyediakan komponen yang ditentukan secara langsung menggunakan skrip penyediaan kustom Anda sendiri, yang dikenal sebagai penyediaan berbasis. CodeBuild Untuk informasi selengkapnya, lihat the section called “CodeBuild penyediaan”.

Komponen di AWS Proton konsol

Gunakan AWS Proton konsol untuk membuat, memperbarui, melihat, dan menggunakan AWS Proton komponen.

Halaman konsol berikut terkait dengan komponen. Kami menyertakan tautan langsung ke halaman konsol tingkat atas.

- Komponen - Lihat daftar komponen di AWS akun Anda. Anda dapat membuat komponen baru, dan memperbarui atau menghapus komponen yang ada. Pilih nama komponen pada daftar untuk melihat halaman detailnya.

  Daftar serupa juga ada di halaman detail Lingkungan dan detail contoh Layanan. Daftar ini hanya menampilkan komponen yang terkait dengan sumber daya yang sedang dilihat. Bila Anda membuat komponen dari salah satu daftar ini, AWS Proton pra-pilih lingkungan terkait pada halaman Create component.
- Detail komponen - Untuk melihat halaman detail komponen, pilih nama komponen pada daftar Komponen.

  Pada halaman detail, lihat detail dan status komponen, dan perbarui atau hapus komponen. Melihat dan mengelola daftar output (misalnya, sumber daya yang disediakan ARNs), CloudFormation tumpukan yang disediakan, dan tag yang ditetapkan.
- Buat komponen - Buat komponen. Masukkan nama dan deskripsi komponen, pilih sumber daya terkait, tentukan file IAc sumber komponen, dan tetapkan tag.

- Perbarui komponen — Untuk memperbarui komponen, pilih komponen pada daftar Komponen, dan kemudian, pada menu Tindakan, pilih Perbarui komponen. Atau, pada halaman detail Komponen, pilih Perbarui.

  Anda dapat memperbarui sebagian besar detail komponen. Anda tidak dapat memperbarui nama komponen. Dan Anda dapat memilih apakah akan menerapkan ulang komponen atau tidak setelah pembaruan berhasil.
- Konfigurasi lingkungan — Saat membuat atau memperbarui lingkungan, Anda dapat menentukan peran Komponen. Peran ini mengontrol kemampuan untuk menjalankan komponen yang ditentukan secara langsung di lingkungan dan memberikan izin untuk menyediakannya.
- Buat versi templat layanan baru — Saat membuat versi templat layanan, Anda dapat menentukan sumber komponen yang didukung untuk versi templat. Ini mengontrol kemampuan untuk melampirkan komponen ke instance layanan layanan berdasarkan versi template ini.

Komponen dalam AWS Proton API dan AWS CLI

Gunakan AWS Proton API atau AWS CLI untuk membuat, memperbarui, melihat, dan menggunakan AWS Proton komponen.

Tindakan API berikut secara langsung mengelola sumber daya AWS Proton komponen.

- CreateComponent— Buat AWS Proton komponen.
- DeleteComponent— Hapus AWS Proton komponen.
- GetComponent— Dapatkan data terperinci untuk suatu komponen.
- ListComponentOutputs— Dapatkan daftar komponen Infrastruktur sebagai output Kode (IAc).
- ListComponentProvisionedResources— Buat daftar sumber daya yang disediakan untuk komponen dengan detail.
- ListComponents— Daftar komponen dengan data ringkasan. Anda dapat memfilter daftar hasil berdasarkan lingkungan, layanan, atau satu contoh layanan.

Tindakan API berikut dari AWS Proton sumber daya lain memiliki beberapa fungsi yang terkait dengan komponen.

- CreateEnvironment, UpdateEnvironment— Gunakan `componentRoleArn` untuk menentukan Nama Sumber Daya Amazon (ARN) dari peran layanan IAM yang AWS Proton digunakan saat

menyediakan komponen yang ditentukan secara langsung di lingkungan ini. Ini menentukan ruang lingkup infrastruktur yang dapat disediakan oleh komponen yang didefinisikan secara langsung.

- CreateServiceTemplateVersion— Gunakan `supportedComponentSources` untuk menentukan sumber komponen yang didukung. Komponen dengan sumber yang didukung dapat dilampirkan ke instance layanan berdasarkan versi template layanan ini.

Pertanyaan yang sering diajukan komponen

Apa siklus hidup suatu komponen?

Komponen dapat berada dalam keadaan terpasang atau terlepas. Mereka dirancang untuk dilampirkan ke instance layanan dan meningkatkan infrastrukturnya sebagian besar waktu. Komponen terpisah berada dalam keadaan transisi yang memungkinkan Anda menghapus komponen atau melampirkannya ke instance layanan lain dengan cara yang terkontrol dan aman. Untuk informasi selengkapnya, lihat the section called “Status komponen”.

Mengapa saya tidak dapat menghapus komponen terlampir saya?

Solusi: Untuk menghapus komponen terlampir, perbarui komponen untuk melepaskannya dari instance layanan, validasi stabilitas instance layanan, dan kemudian hapus komponen tersebut.

Mengapa ini diperlukan? Komponen terlampir menyediakan infrastruktur tambahan yang dibutuhkan aplikasi Anda untuk menjalankan fungsi runtime-nya. Instance layanan mungkin menggunakan output komponen untuk mendeteksi dan menggunakan sumber daya infrastruktur ini. Menghapus komponen, sehingga menghapus sumber daya infrastrukturnya, dapat mengganggu instance layanan terlampir.

Sebagai langkah keamanan tambahan, Anda AWS Proton mengharuskan Anda memperbarui komponen dan melepaskannya dari instance layanannya sebelum Anda dapat menghapusnya. Anda kemudian dapat memvalidasi instance layanan Anda untuk memastikan bahwa instance tersebut terus digunakan dan berfungsi dengan baik. Jika Anda mendeteksi masalah, Anda dapat dengan cepat memasang kembali komponen ke instance layanan, lalu bekerja untuk memperbaiki masalah tersebut. Jika Anda yakin bahwa instance layanan Anda bebas dari ketergantungan apa pun pada komponen, Anda dapat menghapus komponen dengan aman.

Mengapa saya tidak dapat mengubah instance layanan terlampir komponen secara langsung?

Solusi: Untuk mengubah lampiran, perbarui komponen untuk melepaskannya dari instance layanan, validasi stabilitas komponen dan instance layanan, lalu lampirkan komponen ke instance layanan baru.

Mengapa ini diperlukan? Komponen dirancang untuk dilampirkan ke instance layanan. Komponen Anda mungkin menggunakan input instance layanan untuk penamaan dan konfigurasi sumber daya infrastruktur. Mengubah instance layanan terlampir dapat mengganggu komponen (selain kemungkinan gangguan pada instance layanan, seperti yang dijelaskan di FAQ sebelumnya, Mengapa saya tidak dapat menghapus komponen terlampir saya?). Misalnya, ini dapat menyebabkan penggantian nama, dan bahkan mungkin penggantian, sumber daya yang ditentukan dalam template IAc komponen.

Sebagai langkah keamanan tambahan, Anda AWS Proton mengharuskan Anda memperbarui komponen dan melepaskannya dari instance layanannya sebelum Anda dapat melampirkannya ke instance layanan lain. Anda kemudian dapat memvalidasi stabilitas komponen dan instance layanan sebelum melampirkan komponen ke instance layanan baru.

Status komponen

AWS Proton komponen dapat berada dalam dua keadaan yang berbeda secara fundamental:

- Terlampir - Komponen dilampirkan ke instance layanan. Ini mendefinisikan infrastruktur yang mendukung fungsionalitas runtime dari instance layanan. Komponen memperluas infrastruktur yang didefinisikan dalam template lingkungan dan layanan dengan infrastruktur yang ditentukan pengembang.

  Komponen tipikal berada dalam keadaan terlampir di sebagian besar bagian yang berguna dari siklus hidupnya.
- Terpisah — Komponen dikaitkan dengan AWS Proton lingkungan, dan tidak dilampirkan ke instance layanan apa pun di lingkungan.

  Ini adalah keadaan transisi untuk memperpanjang masa pakai komponen di luar satu instance layanan.

Tabel berikut memberikan perbandingan tingkat atas dari status komponen yang berbeda.

	Terlampir	Terlepas
Tujuan utama negara	Untuk memperluas infrastruktur instance layanan.	Untuk menjaga infrastruktur komponen antara lampiran instance layanan.
Terkait dengan	Contoh layanan dan lingkungan	Lingkungan
Properti spesifik kunci	• Nama layanan • Nama contoh layanan • Spesifikasi	• Nama lingkungan
Dapat dihapus	✗ Tidak	✓ Ya
Dapat diperbarui ke instance layanan lain	✗ Tidak	✓ Ya
Dapat membaca input	✓ Ya	✗ Tidak

Tujuan utama komponen adalah untuk dilampirkan ke instance layanan dan memperluas infrastrukturnya dengan sumber daya tambahan. Komponen terlampir dapat membaca input dari instance layanan sesuai dengan spesifikasi. Anda tidak dapat langsung menghapus komponen atau melampirkannya ke instance layanan yang berbeda. Anda juga tidak dapat menghapus instance layanannya atau layanan dan lingkungan terkait. Untuk melakukan hal-hal ini, perbarui komponen untuk melepaskannya dari instance layanannya terlebih dahulu.

Untuk mempertahankan infrastruktur komponen di luar masa pakai satu instance layanan, Anda memperbarui komponen dan melepaskannya dari instance layanannya dengan menghapus nama instance layanan dan layanan. Keadaan terpisah ini adalah keadaan transisi. Komponen tidak memiliki input. Infrastrukturnya tetap disediakan dan Anda dapat memperbaruinya. Anda dapat menghapus sumber daya yang dikaitkan dengan komponen saat dilampirkan (instance layanan, layanan). Anda dapat menghapus komponen atau memperbaruinya untuk dilampirkan ke instance layanan lagi.

Infrastruktur komponen sebagai file kode

Infrastruktur komponen sebagai file kode (IAc) mirip dengan file untuk AWS Proton sumber daya lainnya. Pelajari di sini tentang beberapa detail yang khusus untuk komponen. Untuk informasi selengkapnya tentang pembuatan file IAC AWS Proton, lihat. Penulisan templat dan bundel

Menggunakan parameter dengan komponen

Namespace AWS Proton parameter mencakup beberapa parameter yang dapat dirujuk oleh file layanan IAc untuk mendapatkan nama dan output komponen terkait. Namespace juga menyertakan parameter yang dapat dirujuk oleh file komponen IAc untuk mendapatkan input, output, dan nilai sumber daya dari lingkungan, layanan, dan instance layanan yang dikaitkan dengan komponen tersebut.

Sebuah komponen tidak memiliki input sendiri—ia mendapatkan inputnya dari instance layanan yang dilampirkan. Sebuah komponen juga dapat membaca output lingkungan.

Untuk informasi selengkapnya tentang penggunaan parameter dalam komponen dan file IAC layanan terkait, lihatthe section called “Parameter komponen CloudFormation IAc”. Untuk informasi umum tentang AWS Proton parameter dan referensi lengkap dari namespace parameter, lihat. the section called “Parameter”

Menulis file IAc yang kuat

Sebagai administrator, ketika Anda membuat versi template layanan, Anda dapat memutuskan apakah Anda ingin mengizinkan instance layanan yang dibuat dari versi template untuk memiliki komponen yang dilampirkan. Lihat supportedComponentSourcesparameter aksi CreateServiceTemplateVersionAPI di Referensi AWS Proton API. Namun, untuk setiap instance layanan future, orang yang membuat instance, memutuskan apakah akan melampirkan komponen ke dalamnya atau tidak, dan (dalam kasus komponen yang didefinisikan secara langsung) membuat komponen IAc biasanya orang yang berbeda—pengembang yang menggunakan template layanan Anda. Oleh karena itu, Anda tidak dapat menjamin bahwa komponen akan dilampirkan ke instance layanan. Anda juga tidak dapat menjamin keberadaan nama keluaran komponen tertentu atau validitas dan keamanan nilai output ini.

AWS Proton dan sintaks Jinja membantu Anda mengatasi masalah ini dan membuat templat layanan kuat yang dirender tanpa kegagalan dengan cara berikut:

- AWS Proton filter parameter — Saat Anda merujuk ke properti keluaran komponen, Anda dapat menggunakan filter parameter —pengubah yang memvalidasi, memfilter, dan memformat nilai parameter. Untuk informasi selengkapnya dan contoh tambahan, lihat the section called "CloudFormation filter parameter".
- Default properti tunggal — Bila Anda merujuk ke sumber daya tunggal atau properti output komponen, Anda dapat menjamin bahwa rendering template layanan Anda tidak akan gagal dengan menggunakan `default` filter, dengan atau tanpa nilai default. Jika komponen, atau parameter keluaran tertentu yang Anda maksud, tidak ada, nilai default (atau string kosong, jika Anda belum menentukan nilai default) akan dirender sebagai gantinya, dan rendering berhasil. Untuk informasi selengkapnya, lihat the section called "Berikan nilai default".

  Contoh:

  - `{{ service_instance.components.default.name | default("") }}`
  - `{{ service_instance.components.default.outputs.my-output | default("17") }}`

  > **Note**
  >
  > Jangan bingung `.default` bagian dari namespace, yang menunjuk komponen yang didefinisikan secara langsung, dengan `default` filter, yang memberikan nilai default ketika properti direferensikan tidak ada.

- Referensi seluruh objek — Ketika Anda merujuk ke seluruh komponen, atau ke koleksi output komponen, AWS Proton mengembalikan objek kosong`{}`, dan karenanya menjamin bahwa rendering template layanan Anda tidak akan gagal. Anda tidak perlu menggunakan filter apa pun. Pastikan untuk membuat referensi dalam konteks yang dapat mengambil objek kosong, atau gunakan `{{ if .. }}` kondisi untuk menguji objek kosong.

  Contoh:

  - `{{ service_instance.components.default }}`
  - `{{ service_instance.components.default.outputs }}`

CloudFormation Contoh komponen

Berikut adalah contoh lengkap dari komponen yang didefinisikan AWS Proton secara langsung dan bagaimana Anda dapat menggunakannya dalam suatu AWS Proton layanan. Komponen ini

menyediakan bucket Amazon Simple Storage Service (Amazon S3) dan kebijakan akses terkait. Instance layanan dapat merujuk ke bucket ini dan menggunakannya. Nama bucket didasarkan pada nama lingkungan, layanan, instance layanan, dan komponen, yang berarti bahwa bucket digabungkan dengan instance spesifik dari template komponen yang memperluas instance layanan tertentu. Pengembang dapat membuat beberapa komponen berdasarkan templat komponen ini, untuk menyediakan bucket Amazon S3 untuk berbagai instans layanan dan kebutuhan fungsional.

Contoh ini mencakup penulisan berbagai CloudFormation infrastruktur yang diperlukan sebagai file kode (IAc) dan membuat peran yang diperlukan AWS Identity and Access Management (IAM). Contoh mengelompokkan langkah-langkah dengan peran orang yang memiliki.

Langkah-langkah administrator

Untuk memungkinkan pengembang menggunakan komponen dengan layanan

1. Buat peran AWS Identity and Access Management (IAM) yang mencakup sumber daya yang secara langsung mendefinisikan komponen yang berjalan di lingkungan Anda dapat menyediakan. AWS Proton mengasumsikan peran ini nanti untuk menyediakan komponen yang didefinisikan secara langsung di lingkungan.

   Untuk contoh ini, gunakan kebijakan berikut:

   Example peran komponen yang didefinisikan secara langsung

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:CancelUpdateStack",
        "cloudformation:CreateChangeSet",
        "cloudformation:DeleteChangeSet",
        "cloudformation:DescribeStacks",
        "cloudformation:ContinueUpdateRollback",
        "cloudformation:DetectStackResourceDrift",
        "cloudformation:DescribeStackResourceDrifts",
        "cloudformation:DescribeStackEvents",
        "cloudformation:CreateStack",
```

```
        "cloudformation:DeleteStack",
        "cloudformation:UpdateStack",
        "cloudformation:DescribeChangeSet",
        "cloudformation:ExecuteChangeSet",
        "cloudformation:ListChangeSets",
        "cloudformation:ListStackResources"
      ],
      "Resource": "arn:aws:cloudformation:*:123456789012:stack/AWSProton-*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:CreateBucket",
        "s3:DeleteBucket",
        "s3:GetBucket*",
        "iam:CreatePolicy",
        "iam:DeletePolicy",
        "iam:GetPolicy",
        "iam:ListPolicyVersions",
        "iam:DeletePolicyVersion"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": "cloudformation.amazonaws.com"
        }
      }
    }
  ]
}
```

2. Berikan peran yang Anda buat pada langkah sebelumnya saat Anda membuat atau memperbarui lingkungan. Di AWS Proton konsol, tentukan peran Komponen pada halaman Konfigurasi lingkungan. Jika Anda menggunakan AWS Proton API atau AWS CLI, tentukan `componentRoleArn` tindakan CreateEnvironmentatau UpdateEnvironmentAPI.
3. Buat template layanan yang mengacu pada komponen yang didefinisikan secara langsung yang dilampirkan ke instance layanan.

   Contoh menunjukkan cara menulis template layanan yang kuat yang tidak rusak jika komponen tidak dilampirkan ke instance layanan.

Example layanan file CloudFormation iAc menggunakan komponen

```
# service/instance_infrastructure/cloudformation.yaml

Resources:
  TaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      TaskRoleArn: !Ref TaskRole
      ContainerDefinitions:
        - Name: '{{service_instance.name}}'
          # ...
          {% if service_instance.components.default.outputs | length > 0 %}
          Environment:
            {{ service_instance.components.default.outputs |
                proton_cfn_ecs_task_definition_formatted_env_vars }}
          {% endif %}

  # ...

  TaskRole:
    Type: AWS::IAM::Role
    Properties:
      # ...
      ManagedPolicyArns:
        - !Ref BaseTaskRoleManagedPolicy
        {{ service_instance.components.default.outputs
            | proton_cfn_iam_policy_arns }}

  # Basic permissions for the task
  BaseTaskRoleManagedPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      # ...
```

4. Buat versi minor template layanan baru yang mendeklarasikan komponen yang didefinisikan secara langsung sebagai didukung.

   - Bundel templat di Amazon S3 — Di AWS Proton konsol, saat Anda membuat versi templat layanan, untuk sumber komponen yang didukung, pilih Ditentukan secara langsung. Jika Anda menggunakan AWS Proton API atau AWS CLI, tentukan `DIRECTLY_DEFINED` dalam

`supportedComponentSources` parameter tindakan CreateServiceTemplateVersionatau UpdateServiceTemplateVersionAPI.

- Sinkronisasi templat — Komit perubahan ke repositori bundel templat layanan Anda, tempat Anda menentukan `DIRECTLY_DEFINED` sebagai item `supported_component_sources:` dalam `.template-registration.yaml` file di direktori versi utama. Untuk informasi selengkapnya tentang file ini, lihatthe section called “Menyinkronkan templat layanan”.

5. Publikasikan template layanan baru versi minor. Untuk informasi selengkapnya, lihat the section called “Publikasikan”.
6. Pastikan untuk mengizinkan peran IAM pengembang yang menggunakan template layanan ini. `proton:CreateComponent`

Langkah-langkah pengembang

Untuk menggunakan komponen yang didefinisikan secara langsung dengan instance layanan

1. Buat layanan yang menggunakan versi template layanan yang dibuat administrator dengan dukungan komponen. Atau, perbarui salah satu instance layanan Anda yang ada untuk menggunakan versi template terbaru.
2. Tulis file template komponen iAc yang menyediakan bucket Amazon S3 dan kebijakan akses terkait dan mengekspos sumber daya ini sebagai output.

   Example file komponen CloudFormation iAc

```
# cloudformation.yaml

# A component that defines an S3 bucket and a policy for accessing the bucket.
Resources:
  S3Bucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: '{{environment.name}}-{{service.name}}-{{service_instance.name}}-
{{component.name}}'
  S3BucketAccessPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      PolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Effect: Allow
```

```
            Action:
              - 's3:Get*'
              - 's3:List*'
              - 's3:PutObject'
            Resource: !GetAtt S3Bucket.Arn
Outputs:
  BucketName:
    Description: "Bucket to access"
    Value: !GetAtt S3Bucket.Arn
  BucketAccessPolicyArn:
    Value: !Ref S3BucketAccessPolicy
```

3. Jika Anda menggunakan AWS Proton API atau AWS CLI, tulis file manifes untuk komponen tersebut.

   Example manifes komponen yang didefinisikan secara langsung

   ```
   infrastructure:
     templates:
       - file: "cloudformation.yaml"
         rendering_engine: jinja
         template_language: cloudformation
   ```

4. Buat komponen yang didefinisikan secara langsung. AWS Proton mengasumsikan peran komponen yang didefinisikan administrator untuk menyediakan komponen.

   Di AWS Proton konsol, pada halaman Components, pilih Create component. Untuk pengaturan Komponen, masukkan nama Komponen dan deskripsi Komponen opsional. Untuk lampiran Komponen, pilih Lampirkan komponen ke instance layanan. Pilih contoh lingkungan, layanan, dan layanan Anda. Untuk sumber Komponen, pilih CloudFormation, lalu pilih file komponen IAc.

   > **Note**
   >
   > Anda tidak perlu menyediakan manifes—konsol membuatnya untuk Anda.

   Jika Anda menggunakan AWS Proton API atau AWS CLI, gunakan tindakan CreateComponentAPI. Tetapkan komponen `name` dan opsional`description`. Set`environmentName`,`serviceName`, dan`serviceInstanceName`. Atur `templateSource` dan `manifest` ke jalur file yang Anda buat.

> **Note**
>
> Menentukan nama lingkungan adalah opsional saat Anda menentukan nama instance layanan dan layanan. Kombinasi keduanya unik di AWS akun Anda, dan AWS Proton dapat menentukan lingkungan dari instance layanan.

5. Perbarui instance layanan Anda untuk menerapkannya kembali. AWS Proton menggunakan output dari komponen Anda dalam template instance layanan yang dirender, untuk memungkinkan aplikasi Anda menggunakan bucket Amazon S3 yang disediakan komponen tersebut.

Menggunakan repositori git dengan AWS Proton

AWS Proton menggunakan repositori git untuk berbagai tujuan. Daftar berikut mengkategorikan jenis repositori yang terkait dengan sumber daya. AWS Proton Untuk AWS Proton fitur yang berulang kali terhubung ke repositori Anda untuk mendorong konten ke sana atau menarik konten darinya, Anda harus mendaftarkan tautan repositori di akun Anda. AWS Proton AWS Tautan repositori adalah seperangkat properti yang AWS Proton dapat digunakan ketika terhubung ke repositori. AWS Proton saat ini mendukung GitHub, GitHub Enterprise, dan BitBucket.

Repositori pengembang

Repositori kode — Repositori yang digunakan pengembang untuk menyimpan kode aplikasi. Digunakan untuk penyebaran kode. AWS Proton tidak berinteraksi langsung dengan repositori ini. Ketika pengembang menyediakan layanan yang menyertakan pipeline, mereka menyediakan nama repositori dan cabang untuk membaca kode aplikasi mereka. AWS Proton meneruskan informasi ini ke pipa yang disediakannya.

Untuk informasi selengkapnya, lihat the section called “Buat”.

Repositori administrator

Template repository — Sebuah repositori tempat administrator menyimpan bundel template. AWS Proton Digunakan untuk sinkronisasi template. Ketika administrator membuat template di AWS Proton, mereka dapat menunjuk ke repositori template, dan AWS Proton membuat template baru tetap sinkron dengannya. Ketika administrator memperbarui bundel template di repositori, AWS Proton secara otomatis membuat versi template baru. Tautkan repositori template AWS Proton sebelum Anda dapat menggunakannya untuk sinkronisasi.

Untuk informasi selengkapnya, lihat the section called “Konfigurasi sinkronisasi templat”.

> **Note**
>
> Repositori template tidak diperlukan jika Anda terus mengunggah template Anda ke Amazon Simple Storage Service (Amazon S3) dan memanggil AWS Proton APIs manajemen template untuk membuat template atau versi template baru.

Repositori penyediaan yang dikelola sendiri

Infrastructure repository — Sebuah repositori yang menampung template infrastruktur yang dirender. Digunakan untuk penyediaan infrastruktur sumber daya yang dikelola sendiri. Ketika administrator membuat lingkungan untuk penyediaan yang dikelola sendiri, mereka menyediakan repositori. AWS Proton mengirimkan pull requests (PRs) ke repositori ini untuk membuat infrastruktur untuk lingkungan dan untuk setiap instance layanan yang diterapkan ke lingkungan. Tautkan repositori infrastruktur AWS Proton sebelum Anda dapat menggunakannya untuk penyediaan infrastruktur yang dikelola sendiri.

Pipeline repository — Sebuah repositori yang digunakan untuk membuat pipeline. Digunakan untuk penyediaan jaringan pipa yang dikelola sendiri. Menggunakan repositori tambahan untuk menyediakan saluran pipa memungkinkan AWS Proton untuk menyimpan konfigurasi pipa secara independen dari lingkungan atau layanan individu. Anda hanya perlu menyediakan satu repositori pipeline untuk semua layanan penyediaan yang dikelola sendiri. Tautkan repositori pipeline AWS Proton sebelum Anda dapat menggunakannya untuk penyediaan pipa yang dikelola sendiri.

Untuk informasi selengkapnya, lihat the section called “AWS-penyediaan terkelola”.

Topik

- Buat tautan ke repositori Anda
- Lihat data repositori tertaut
- Hapus tautan repositori

Buat tautan ke repositori Anda

Anda dapat membuat tautan ke repositori Anda menggunakan konsol atau CLI. Saat Anda membuat tautan repositori, AWS Proton buat peran terkait layanan untuk Anda.

Konsol Manajemen AWS

Buat tautan ke repositori Anda seperti yang ditunjukkan pada langkah-langkah konsol berikut.

1. Di AWS Proton konsol, pilih Repositori.
2. Pilih Buat repositori.
3. Di halaman Tautan repositori baru, di bagian Detail Repositori:

a. Pilih penyedia repositori Anda.

b. Pilih salah satu koneksi yang ada. Jika Anda tidak memilikinya, pilih Tambahkan CodeStar koneksi baru untuk membuat koneksi, lalu kembali ke AWS Proton konsol, segarkan daftar koneksi, dan pilih koneksi baru Anda.

c. Pilih dari repositori kode sumber Anda yang terhubung.

4. [opsional] Di bagian Tag, pilih Tambahkan tag baru satu kali atau beberapa kali, lalu masukkan pasangan Kunci dan Nilai.
5. Pilih Buat repositori.
6. Lihat data detail untuk repositori tertaut Anda.

AWS CLI

Buat dan daftarkan tautan ke repositori Anda.

Jalankan perintah berikut:

```
$ aws proton create-repository \
    --name myrepos/environments \
    --connection-arn "arn:aws:codestar-connections:region-
id:123456789012:connection/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111" \
    --provider "GITHUB" \
    --encryption-key "arn:aws:kms:region-id:123456789012:key/bPxRfiCYEXAMPLEKEY" \
    --tags key=mytag1,value=value1 key=mytag2,value=value2
```

Dua parameter terakhir, --encryption-key dan--tags, adalah opsional.

Respons:

```
{
    "repository": {
        "arn": "arn:aws:proton:region-id:123456789012:repository/github:myrepos/
environments",
        "connectionArn": "arn:aws:codestar-connections:region-
id:123456789012:connection/2ad03b28-a7c4-EXAMPLE11111",
        "encryptionKey": "arn:aws:kms:region-id:123456789012:key/
bPxRfiCYEXAMPLEKEY",
        "name": "myrepos/environments",
        "provider": "GITHUB"
    }
```

```
}
```

Setelah Anda membuat link repositori, Anda dapat melihat daftar AWS dan tag terkelola pelanggan, seperti yang ditunjukkan dalam perintah contoh berikut. AWS Proton secara otomatis menghasilkan tag AWS terkelola untuk Anda. Anda juga dapat memodifikasi dan membuat tag yang dikelola pelanggan menggunakan AWS CLI. Untuk informasi selengkapnya, lihat AWS Proton sumber daya dan penandaan.

Perintah:

```
$ aws proton list-tags-for-resource \
    --resource-arn "arn:aws:proton:region-id:123456789012:repository/github:myrepos/environments"
```

Lihat data repositori tertaut

Anda dapat membuat daftar dan melihat detail repositori tertaut menggunakan konsol atau file. AWS CLI Untuk tautan repositori yang digunakan untuk menyinkronkan repositori git dengan AWS Proton, Anda dapat mengambil definisi dan status sinkronisasi repositori menggunakan. AWS CLI

Konsol Manajemen AWS

Daftar dan lihat detail repositori tertaut menggunakan konsol.AWS Proton

1. Untuk daftar repositori tertaut Anda, pilih Repositori di panel navigasi.
2. Untuk melihat data detail, pilih nama repositori.

AWS CLI

Buat daftar repositori tertaut Anda.

Jalankan perintah berikut:

```
$ aws proton list-repositories
```

Respons:

```
{
    "repositories": [
```

```
        {
            "arn": "arn:aws:proton:region-id:123456789012:repository/github:myrepos/
templates",
            "name": "myrepos/templates",
            "provider": "GITHUB"
        },
        {
            "arn": "arn:aws:proton:region-id:123456789012:repository/github:myrepos/
environments",
            "name": "myrepos/environments",
            "provider": "GITHUB"
        }
    ]
}
```

Lihat detail repositori tertaut.

Jalankan perintah berikut:

```
$ aws proton get-repository \
    --name myrepos/templates \
    --provider "GITHUB"
```

Respons:

```
{
    "repository": {
        "arn": "arn:aws:proton:region-id:123456789012:repository/github:myrepos/
templates",
        "name": "myrepos/templates",
        "provider": "GITHUB"
    }
}
```

Buat daftar repositori yang disinkronkan.

Contoh berikut mencantumkan repositori yang Anda konfigurasikan untuk sinkronisasi templat.

Jalankan perintah berikut:

```
$ aws proton list-repository-sync-definitions \
    --branch "main" \
```

```
    --repository-name myrepos/templates \
    --repository-provider "GITHUB" \
    --sync-type "TEMPLATE_SYNC"
```

Lihat status sinkronisasi repositori.

Contoh berikut mengambil status sinkronisasi dari repositori sinkronisasi template.

Jalankan perintah berikut:

```
$ aws proton get-repository-sync-status \
    --branch "main" \
    --repository-name myrepos/templates \
    --repository-provider "GITHUB" \
    --sync-type "TEMPLATE_SYNC"
```

Respons:

```
{
    "latestSync": {
        "events": [
            {
                "event": "Clone started",
                "time": "2021-11-21T00:26:35.883000+00:00",
                "type": "CLONE_STARTED"
            },
            {
                "event": "Updated configuration",
                "time": "2021-11-21T00:26:41.894000+00:00",
                "type": "CONFIG_UPDATED"
            },
            {
                "event": "Starting syncs for commit 62c03ff86eEXAMPLE1111111",
                "externalId": "62c03ff86eEXAMPLE1111111",
                "time": "2021-11-21T00:26:44.861000+00:00",
                "type": "STARTING_SYNC"
            }
        ],
        "startedAt": "2021-11-21T00:26:29.728000+00:00",
        "status": "SUCCEEDED"
    }
}
```

Hapus tautan repositori

Anda dapat menghapus tautan repositori dengan menggunakan konsol atau file. AWS CLI

> **Note**
>
> Menghapus tautan repositori hanya menghapus tautan terdaftar yang ada AWS Proton di akun Anda. AWS Itu tidak menghapus informasi apa pun dari repositori Anda.

Konsol Manajemen AWS

Hapus tautan repositori menggunakan konsol.

Di halaman detail repositori.

1. Di AWS Proton konsol, pilih Repositori.
2. Dalam daftar repositori, pilih tombol radio di sebelah kiri repositori yang ingin Anda hapus.
3. Pilih Hapus.
4. Modal meminta Anda untuk mengonfirmasi tindakan Hapus.
5. Ikuti instruksi dan pilih Ya, hapus.

AWS CLI

Hapus tautan repositori.

Jalankan perintah berikut:

```
$ aws proton delete-repository \
    --name myrepos/templates \
    --provider"GITHUB"
```

Respons:

```
{
    "repository": {
        "arn": "arn:aws:proton:region-id:123456789012:repository/github:myrepos/
templates",
        "name": "myrepos/templates",
```

```
        "provider": "GITHUB"
    }
}
```

Pemantauan AWS Proton

Pemantauan adalah bagian penting dari menjaga keandalan, ketersediaan, dan kinerja AWS Proton dan AWS solusi Anda. Bagian berikut menjelaskan alat pemantauan yang dapat Anda gunakan AWS Proton.

Otomatisasi dengan AWS Proton EventBridge

Anda dapat memantau AWS Proton acara di Amazon EventBridge. EventBridge memberikan aliran data real-time dari aplikasi Anda sendiri, aplikasi software-as-a-service (SaaS), dan. Layanan AWS Anda dapat mengonfigurasi peristiwa untuk merespons perubahan status AWS sumber daya. EventBridge merutekan data ini kemudian ke layanan target seperti AWS Lambda dan Amazon Simple Notification Service. Peristiwa ini sama dengan yang muncul di CloudWatch Acara Amazon. CloudWatch Peristiwa memberikan aliran peristiwa sistem yang mendekati waktu nyata yang menggambarkan perubahan AWS sumber daya. Untuk informasi selengkapnya, lihat Apa itu Amazon EventBridge? di Panduan EventBridge Pengguna Amazon.

Gunakan EventBridge untuk diberitahu tentang perubahan status dalam alur kerja AWS Proton penyediaan.

Tipe peristiwa

Acara terdiri dari aturan yang mencakup pola acara dan target. Anda mengonfigurasi aturan dengan memilih pola acara dan objek target:

Pola peristiwa

> Setiap aturan dinyatakan sebagai pola peristiwa dengan sumber dan jenis peristiwa untuk memantau dan target acara. Untuk memantau peristiwa, Anda membuat aturan dengan layanan yang Anda pantau sebagai sumber acara. Misalnya, Anda dapat membuat aturan dengan pola peristiwa yang digunakan AWS Proton sebagai sumber peristiwa untuk memicu aturan ketika ada perubahan dalam status penerapan.

Target

> Aturan menerima layanan yang dipilih sebagai target acara. Anda dapat mengatur layanan target untuk mengirim pemberitahuan, menangkap informasi status, mengambil tindakan korektif, memulai acara, atau mengambil tindakan lain.

Objek acara berisi bidang standar ID, akun, tipe detail AWS Region, sumber, versi, sumber daya, waktu (opsional). Bidang detail adalah objek bersarang yang berisi bidang khusus untuk acara tersebut.

AWS Proton peristiwa dipancarkan atas dasar upaya terbaik. Penyampaian upaya terbaik berarti bahwa layanan mencoba mengirim semua acara ke EventBridge, tetapi dalam beberapa kasus yang jarang terjadi suatu peristiwa mungkin tidak disampaikan.

Untuk setiap AWS Proton sumber daya yang dapat memancarkan peristiwa, tabel berikut mencantumkan nilai tipe detail, bidang detail, dan (jika tersedia) referensi ke daftar nilai untuk bidang dan detail. `status` `previousStatus` Ketika sumber daya dihapus, nilai bidang `status` detail adalah`DELETED`.

Sumber daya	Nilai tipe detail	Bidang detail

`EnvironmentTemplate`	AWS Proton Perubahan Status Template Lingkungan	`name`
		`status`
		`previousStatus`

`EnvironmentTemplateVersion`	AWS Proton Perubahan Status Versi Template Lingkungan	`name`
		`majorVersion`
		`minorVersion`
		`status`
		`previousStatus`
		nilai status

`ServiceTemplate`	AWS Proton Perubahan Status Template Layanan	`name` `status`

Sumber daya	Nilai tipe detail	Bidang detail
		previousStatus
ServiceTemplateVersion	AWS Proton Perubahan Status Versi Template Layanan	name majorVersion minorVersion status previousStatus nilai status
Environment	AWS Proton Perubahan Status Lingkungan	name status previousStatus
Service	AWS Proton Perubahan Status Layanan	name status previousStatus nilai status

Sumber daya	Nilai tipe detail	Bidang detail

`ServiceInstance`	AWS Proton Perubahan Status Instans Layanan	`name`
		`serviceName`
		`status`
		`previousStatus`

`ServicePipeline`	AWS Proton Perubahan Status Pipa Layanan	`serviceName`
		`status`
		`previousStatus`

`EnvironmentAccountConnection`	AWS Proton Perubahan Status Koneksi Akun Lingkungan	`id`
		`status`
		`previousStatus`
		nilai status

`Component`	AWS Proton Perubahan Status Komponen	`name`
		`status`
		`previousStatus`

AWS Proton contoh acara

Contoh berikut menunjukkan cara-cara yang AWS Proton dapat mengirim acara ke EventBridge.

Template layanan

```
{
```

```
    "source": "aws.proton",
    "detail-type": ["AWS Proton Service Template Status Change"],
    "time": "2021-03-22T23:21:40.734Z",
    "resources": ["arn:aws:proton:region_id:123456789012:service-template/sample-
service-template-name"],
    "detail": {
        "name": "sample-service-template-name",
        "status": "PUBLISHED",
        "previousStatus": "DRAFT"
    }
}
```

Versi template layanan

```
{
    "source": "aws.proton",
    "detail-type": ["AWS Proton Service Template Version Status Change"],
    "time": "2021-03-22T23:21:40.734Z",
    "resources": ["arn:aws:proton:region_id:123456789012:service-template/sample-
service-template-name:1.0"],
    "detail": {
        "name": "sample-service-template-name",
        "majorVersion": "1",
        "minorVersion": "0",
        "status": "REGISTRATION_FAILED",
        "previousStatus": "REGISTRATION_IN_PROGRESS"
    }
}
```

Lingkungan

```
{
    "source": "aws.proton",
    "detail-type": ["AWS Proton Environment Status Change"],
    "time": "2021-03-22T23:21:40.734Z",
    "resources": ["arn:aws:proton:region_id:123456789012:environment/sample-
environment"],
    "detail": {
        "name": "sample-environment",
        "status": "DELETE_FAILED",
        "previousStatus": "DELETE_IN_PROGRESS"
    }
```

```
}
```

EventBridgeTutorial: Kirim peringatan Amazon Simple Notification Service untuk AWS Proton perubahan status layanan

Dalam tutorial ini, Anda menggunakan aturan peristiwa AWS Proton pra-konfigurasi yang menangkap perubahan status untuk layanan Anda AWS Proton . EventBridgemengirimkan perubahan status ke topik Amazon SNS. Anda berlangganan topik dan Amazon SNS mengirimi Anda email perubahan status untuk layanan Anda AWS Proton .

Prasyarat

Anda memiliki AWS Proton layanan yang sudah ada dengan `Active` status. Sebagai bagian dari tutorial ini, Anda dapat menambahkan instance layanan ke layanan ini, dan kemudian menghapus instance.

Jika Anda perlu membuat AWS Proton layanan, lihatMemulai. Untuk informasi selengkapnya, lihat AWS Proton kuota dan the section called “Sunting”.

Langkah 1: Buat dan berlangganan ke topik Amazon SNS

Buat topik Amazon SNS untuk dijadikan target acara untuk aturan acara yang Anda buat di Langkah 2.

Membuat topik Amazon SNS

1. Masuk dan buka konsol Amazon SNS.
2. Di panel navigasi, pilih Topik, Buat topik.
3. Di halaman Buat topik:
    a. Pilih Jenis Standar.
    b. Untuk Nama, masukkan **`tutorial-service-status-change`** dan pilih Buat topik.
4. Di halaman tutorial-service-status-changedetail, pilih Buat langganan.
5. Di halaman Buat langganan:
    a. Untuk Protokol, pilih Email.
    b. UntukTitik akhir, masukkan alamat email yang Anda dapat mengaksesnya dan pilih Buat langganan.

6. Periksa akun email Anda dan tunggu untuk menerima pesan email konfirmasi langganan. Saat Anda menerimanya, buka dan pilih Konfirmasi berlangganan.

Langkah 2: Mendaftarkan aturan peristiwa

Daftarkan aturan acara yang menangkap perubahan status untuk AWS Proton layanan Anda. Untuk informasi selengkapnya, lihat Prasyarat.

Buat aturan acara.

1. Buka EventBridge konsol Amazon.
2. Di panel navigasi, pilih Peristiwa, Aturan.
3. Di halaman Aturan, di bagian Aturan, pilih Buat aturan.
4. Di halaman Buat aturan:

    a. Di bagian Nama dan deskripsi, untuk Nama, masukkan**`tutorial-rule`**.

    b. Di bagian Tentukan pola, pilih Pola acara.

        i. Untuk pola pencocokan Event, pilih Predefined by service.

        ii. Untuk Penyedia layanan, pilih AWS.

        iii. Untuk nama Layanan, pilih AWS Proton.

        iv. Untuk jenis Acara, pilih Perubahan Status AWS Proton Layanan.

        Pola acara muncul di editor teks.

        v. Buka konsol AWS Proton.

        vi. Pada panel navigasi, silakan pilih Layanan.

        vii. Di halaman Layanan, pilih nama AWS Proton layanan Anda.

        viii. Di halaman detail Layanan, salin layanan Amazon Resource Name (ARN).

        ix. Arahkan kembali ke EventBridge konsol dan aturan tutorial Anda dan pilih Edit di editor teks.

        x. Di editor teks, untuk`"resources":`, masukkan layanan ARN yang Anda salin di langkah viii.

```
{
    "source": ["aws.proton"],
    "detail-type": ["AWS Proton Service Status Change"],
```

```
    "resources": ["arn:aws:proton:region-id:123456789012:service/your-
service"]
}
```

xi. Simpan pola acara.

c. Di bagian Pilih target:

i. Untuk Target, pilih topik SNS.

ii. Untuk Topik, pilih tutorial-service-status-change.

d. Pilih Buat.

Langkah 3: Uji aturan acara Anda

Verifikasi bahwa aturan acara Anda berfungsi dengan menambahkan instance ke AWS Proton layanan Anda.

1. Beralih ke AWS Proton konsol.
2. Pada panel navigasi, silakan pilih Layanan.
3. Di halaman Layanan, pilih nama layanan Anda.
4. Di halaman Detail Layanan, pilih Edit.
5. Di halaman Konfigurasi layanan, pilih Berikutnya.
6. Di halaman Konfigurasi pengaturan kustom, di bagian Instans layanan, pilih Tambahkan instance baru.
7. Lengkapi formulir untuk instans Baru Anda:

   a. Masukkan Nama untuk instance baru Anda.

   b. Pilih lingkungan kompatibel yang sama dengan yang Anda pilih untuk instans yang ada.

   c. Masukkan nilai untuk input yang diperlukan.

   d. Pilih Berikutnya.
8. Tinjau input Anda dan pilih Perbarui.
9. Setelah status Layanan`Active`, periksa email Anda untuk memverifikasi bahwa Anda menerima AWS pemberitahuan yang memberikan pembaruan status.

```
{
    "version": "0",
    "id": "af76c382-2b3c-7a0a-cf01-936dff228276",
```

```
    "detail-type": "AWS Proton Service Status Change",
    "source": "aws.proton",
    "account": "123456789012",
    "time": "2021-06-29T20:40:16Z",
    "region": "region-id",
    "resources": ["arn:aws:proton:region-id:123456789012:service/your-service"],
    "detail": {
        "previousStatus": "ACTIVE",
        "status": "UPDATE_IN_PROGRESS",
        "name": "your-service"
    }
}
```

```
{
    "version": "0",
    "id": "87131e29-ad95-bda2-cd30-0ce825dfb0cd",
    "detail-type": "AWS Proton Service Status Change",
    "source": "aws.proton",
    "account": "123456789012",
    "time": "2021-06-29T20:42:27Z",
    "region": "region-id",
    "resources": ["arn:aws:proton:region-id:123456789012:service/your-service"],
    "detail": {
        "previousStatus": "UPDATE_IN_PROGRESS",
        "status": "ACTIVE",
        "name": "your-service"
    }
}
```

Langkah 4: Membersihkan

Hapus topik dan langganan Amazon SNS Anda dan hapus aturan Anda EventBridge .

Hapus topik dan langganan Amazon SNS Anda.

1. Arahkan ke konsol Amazon SNS.
2. Di panel navigasi, pilih Langganan.
3. Di halaman Langganan, pilih langganan yang Anda buat untuk topik bernama `tutorial-service-status-change` lalu pilih Hapus.
4. Di panel navigasi, pilih Topik.

5. Di halaman Topik, pilih topik bernama `tutorial-service-status-change` lalu pilih Hapus.
6. Modal meminta Anda untuk memverifikasi penghapusan. Ikuti instruksi dan pilih Hapus.

Hapus EventBridge aturan Anda.

1. Arahkan ke EventBridge konsol Amazon.
2. Di panel navigasi, pilih Peristiwa, Aturan.
3. Di halaman Aturan, pilih aturan bernama `tutorial-rule` lalu pilih Hapus.
4. Modal meminta Anda untuk memverifikasi penghapusan. Pilih Hapus.

Hapus instance layanan yang ditambahkan.

1. Navigasikan ke konsol AWS Proton tersebut.
2. Pada panel navigasi, silakan pilih Layanan.
3. Di halaman Layanan, pilih nama layanan Anda.
4. Di halaman Detail layanan, pilih Edit dan kemudian Berikutnya.
5. Di halaman Konfigurasi pengaturan kustom, di bagian Contoh layanan, pilih Hapus untuk instance layanan yang Anda buat sebagai bagian dari tutorial ini dan kemudian pilih Berikutnya.
6. Tinjau input Anda dan pilih Perbarui.
7. Modal meminta Anda untuk memverifikasi penghapusan. Ikuti instruksi dan pilih Ya, hapus.

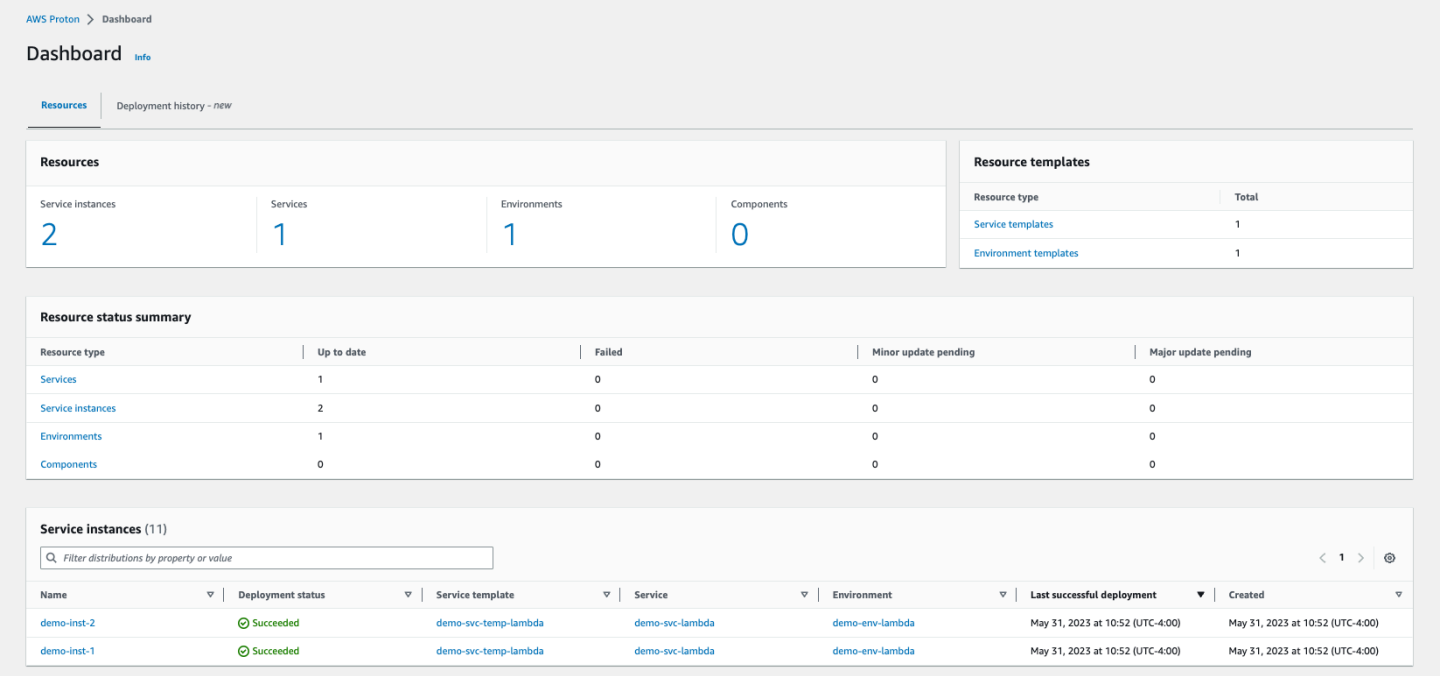
Perbarui infrastruktur dengan AWS Proton dasbor

AWS Proton Dasbor menyediakan ringkasan AWS Proton sumber daya di AWS akun Anda, dengan fokus khusus pada kebuntuan —bagaimana sumber daya yang diterapkan diperbarui. Sumber daya yang digunakan diperbarui saat menggunakan versi yang direkomendasikan dari template terkait. Sumber daya yang out-of-date digunakan mungkin memerlukan pembaruan versi template mayor atau minor.

Lihat dasbor di AWS Proton konsol

Untuk melihat AWS Proton dasbor, buka AWS Proton konsol, lalu, di panel navigasi, pilih Dasbor.

Sumber daya

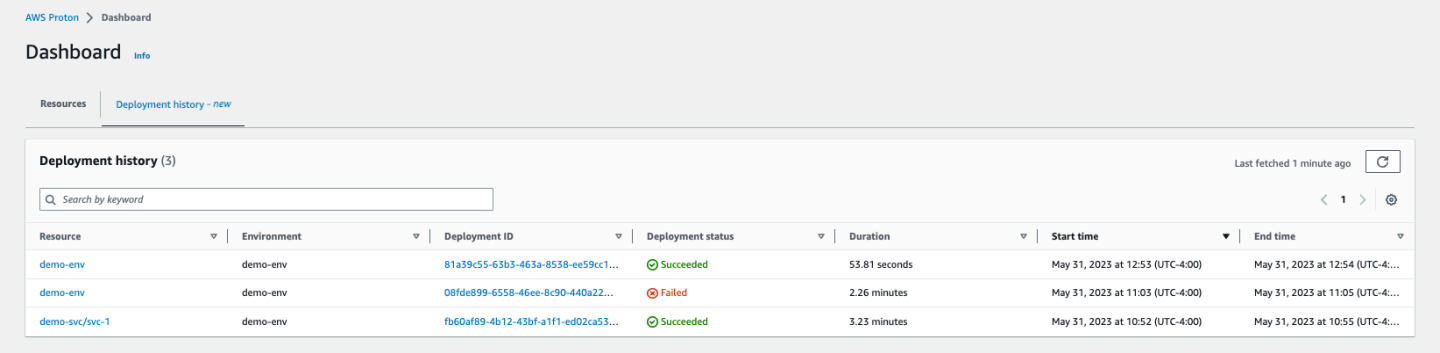


Tab pertama dasbor menampilkan jumlah semua sumber daya di akun Anda. Tab sumber daya menunjukkan jumlah instans layanan, layanan, lingkungan, dan komponen, serta templat sumber daya Anda. Ini juga memecah jumlah sumber daya untuk setiap jenis sumber daya yang diterapkan berdasarkan status sumber daya jenis itu. Tabel instance layanan menunjukkan detail setiap instance layanan—status penerapannya, AWS Proton sumber daya yang terkait dengannya, pembaruan yang tersedia untuknya, dan beberapa stempel waktu.

Anda dapat memfilter daftar instance layanan dengan properti tabel apa pun. Misalnya, Anda dapat memfilter untuk melihat instance layanan dengan penerapan dalam jangka waktu tertentu, atau instance layanan yang kedaluwarsa relatif terhadap rekomendasi versi mayor atau minor.

Pilih nama instance layanan untuk menavigasi ke halaman detail instance layanan, tempat Anda dapat bertindak untuk membuat pembaruan versi yang sesuai. Pilih nama AWS Proton sumber daya lain untuk menavigasi ke halaman detailnya, atau pilih jenis sumber daya untuk menavigasi ke daftar sumber daya masing-masing.

Riwayat penyebaran



Tab riwayat penerapan memungkinkan Anda melihat detail tentang penerapan Anda. Dalam tabel riwayat penerapan, Anda dapat melacak status penerapan, serta ID lingkungan dan penerapan. Anda dapat memilih nama sumber daya atau ID penyebaran untuk melihat detail lebih lanjut, seperti pesan status penerapan dan output sumber daya. Tabel ini juga memungkinkan Anda untuk memfilter pada properti tabel apa pun.

Keamanan di AWS Proton

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 dari cloud dan keamanan dalam cloud:

- Keamanan cloud — AWS bertanggung jawab untuk melindungi infrastruktur yang berjalan Layanan AWS di dalamnya 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 Proton, lihat Layanan AWS di Lingkup oleh Program Kepatuhan Layanan AWS .
- Keamanan di cloud — Tanggung jawab Anda ditentukan oleh Layanan AWS 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 AWS Proton. Topik berikut menunjukkan cara mengonfigurasi AWS Proton untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga belajar cara menggunakan Layanan AWS yang lain yang membantu Anda memantau dan mengamankan AWS Proton sumber daya Anda.

Topik

- Identity and Access Management untuk AWS Proton
- Analisis konfigurasi dan kerentanan di AWS Proton
- Perlindungan data di AWS Proton
- Keamanan infrastruktur di AWS Proton
- Penebangan dan pemantauan di AWS Proton
- Ketahanan di AWS Proton
- Praktik terbaik keamanan untuk AWS Proton
- Pencegahan "confused deputy" lintas layanan
- CodeBuild penyediaan dukungan VPC Amazon khusus

Identity and Access Management untuk AWS Proton

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 diberi wewenang (memiliki izin) untuk menggunakan sumber daya. AWS Proton IAM adalah Layanan AWS yang dapat Anda gunakan tanpa biaya tambahan.

Topik

- Audiens
- Mengautentikasi dengan identitas
- Mengelola akses menggunakan kebijakan
- Bagaimana AWS Proton bekerja dengan IAM
- Contoh kebijakan untuk AWS Proton
- AWS kebijakan terkelola untuk AWS Proton
- Menggunakan peran terkait layanan untuk AWS Proton
- Memecahkan masalah AWS Proton identitas dan akses

Audiens

Cara Anda menggunakan AWS Identity and Access Management (IAM) berbeda berdasarkan peran Anda:

- Pengguna layanan - minta izin dari administrator Anda jika Anda tidak dapat mengakses fitur (lihat Memecahkan masalah AWS Proton identitas dan akses)
- Administrator layanan - tentukan akses pengguna dan mengirimkan permintaan izin (lihat Bagaimana AWS Proton bekerja dengan IAM)
- Administrator IAM - tulis kebijakan untuk mengelola akses (lihat Contoh kebijakan berbasis identitas untuk AWS Proton)

Mengautentikasi dengan identitas

Otentikasi adalah cara Anda masuk AWS menggunakan kredensyal identitas Anda. Anda harus diautentikasi sebagai Pengguna root akun AWS, pengguna IAM, atau dengan mengasumsikan peran IAM.

Anda dapat masuk sebagai identitas federasi menggunakan kredensional dari sumber identitas seperti AWS IAM Identity Center (Pusat Identitas IAM), otentikasi masuk tunggal, atau kredensional. Google/Facebook Untuk informasi selengkapnya tentang cara masuk, lihat Cara masuk ke Akun AWS Anda dalam Panduan Pengguna AWS Sign-In .

Untuk akses terprogram, AWS sediakan SDK dan CLI untuk menandatangani permintaan secara kriptografis. Untuk informasi selengkapnya, lihat AWS Signature Version 4 untuk permintaan API dalam Panduan Pengguna IAM.

Akun AWS pengguna root

Saat Anda membuat Akun AWS, Anda mulai dengan satu identitas masuk yang disebut pengguna Akun AWS root yang memiliki akses lengkap ke semua Layanan AWS dan sumber daya. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari. Untuk tugas yang memerlukan kredensial pengguna root, lihat Tugas yang memerlukan kredensial pengguna root dalam Panduan Pengguna IAM.

Identitas terfederasi

Sebagai praktik terbaik, mewajibkan pengguna manusia untuk menggunakan federasi dengan penyedia identitas untuk mengakses Layanan AWS menggunakan kredensi sementara.

Identitas federasi adalah pengguna dari direktori perusahaan Anda, penyedia identitas web, atau Directory Service yang mengakses Layanan AWS menggunakan kredensi dari sumber identitas. Identitas terfederasi mengambil peran yang memberikan kredensial sementara.

Untuk manajemen akses terpusat, kami menyarankan AWS IAM Identity Center. Untuk informasi selengkapnya, lihat Apa itu Pusat Identitas IAM? dalam Panduan Pengguna AWS IAM Identity Center .

Pengguna dan grup IAM

Pengguna IAM adalah identitas dengan izin khusus untuk satu orang atau aplikasi. Sebaiknya gunakan kredensial sementara alih-alih pengguna IAM dengan kredensial jangka panjang. Untuk informasi selengkapnya, lihat Mewajibkan pengguna manusia untuk menggunakan federasi dengan penyedia identitas untuk mengakses AWS menggunakan kredensi sementara di Panduan Pengguna IAM.

Grup IAM menentukan kumpulan pengguna IAM dan mempermudah pengelolaan izin untuk pengguna dalam jumlah besar. Untuk mempelajari selengkapnya, lihat Kasus penggunaan untuk pengguna IAM dalam Panduan Pengguna IAM.

Peran IAM

Peran IAM adalah identitas dengan izin khusus yang menyediakan kredensial sementara. Anda dapat mengambil peran dengan beralih dari pengguna ke peran IAM (konsol) atau dengan memanggil operasi AWS CLI atau AWS API. Untuk informasi selengkapnya, lihat Metode untuk mengambil peran dalam Panduan Pengguna IAM.

Peran IAM berguna untuk akses pengguna terfederasi, izin pengguna IAM sementara, akses lintas akun, akses lintas layanan, dan aplikasi yang berjalan di Amazon EC2. Untuk informasi selengkapnya, lihat Akses sumber daya lintas akun di IAM dalam Panduan Pengguna IAM.

Mengelola akses menggunakan kebijakan

Anda mengontrol akses AWS dengan membuat kebijakan dan melampirkannya ke AWS identitas atau sumber daya. Kebijakan menentukan izin saat dikaitkan dengan identitas atau sumber daya. AWS mengevaluasi kebijakan ini ketika kepala sekolah membuat permintaan. Sebagian besar kebijakan disimpan AWS sebagai dokumen JSON. Untuk informasi selengkapnya tentang dokumen kebijakan JSON, lihat Gambaran umum kebijakan JSON dalam Panduan Pengguna IAM.

Menggunakan kebijakan, administrator menentukan siapa yang memiliki akses ke apa dengan mendefinisikan principal mana yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Secara default, pengguna dan peran tidak memiliki izin. Administrator IAM membuat kebijakan IAM dan menambahkannya ke peran, yang kemudian dapat diambil oleh pengguna. Kebijakan IAM mendefinisikan izin terlepas dari metode yang Anda gunakan untuk melakukan operasinya.

Kebijakan berbasis identitas

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang Anda lampirkan ke identitas (pengguna, grup, atau peran). Kebijakan ini mengontrol tindakan apa yang bisa dilakukan oleh identitas tersebut, terhadap sumber daya yang mana, dan dalam kondisi apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Tentukan izin IAM kustom dengan kebijakan yang dikelola pelanggan dalam Panduan Pengguna IAM.

Kebijakan berbasis identitas dapat berupa kebijakan inline (disematkan langsung ke dalam satu identitas) atau kebijakan terkelola (kebijakan mandiri yang dilampirkan pada banyak identitas). Untuk mempelajari cara memilih antara kebijakan terkelola dan kebijakan inline, lihat Pilih antara kebijakan terkelola dan kebijakan inline dalam Panduan Pengguna IAM.

Kebijakan berbasis sumber daya

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contohnya termasuk kebijakan kepercayaan peran IAM dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Anda harus menentukan principal dalam kebijakan berbasis sumber daya.

Kebijakan berbasis sumber daya merupakan kebijakan inline yang terletak di layanan tersebut. Anda tidak dapat menggunakan kebijakan AWS terkelola dari IAM dalam kebijakan berbasis sumber daya.

Jenis-jenis kebijakan lain

AWS mendukung jenis kebijakan tambahan yang dapat menetapkan izin maksimum yang diberikan oleh jenis kebijakan yang lebih umum:

- Batasan izin – Menetapkan izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas kepada entitas IAM. Untuk informasi selengkapnya, lihat Batasan izin untuk entitas IAM dalam Panduan Pengguna IAM.
- Kebijakan kontrol layanan (SCPs) — Tentukan izin maksimum untuk organisasi atau unit organisasi di AWS Organizations. Untuk informasi selengkapnya, lihat Kebijakan kontrol layanan dalam Panduan Pengguna AWS Organizations .
- Kebijakan kontrol sumber daya (RCPs) — Tetapkan izin maksimum yang tersedia untuk sumber daya di akun Anda. Untuk informasi selengkapnya, lihat Kebijakan kontrol sumber daya (RCPs) di Panduan AWS Organizations Pengguna.
- Kebijakan sesi – Kebijakan lanjutan yang diteruskan sebagai parameter saat membuat sesi sementara untuk peran atau pengguna terfederasi. Untuk informasi selengkapnya, lihat Kebijakan sesi dalam Panduan Pengguna IAM.

Berbagai jenis kebijakan

Ketika beberapa jenis kebijakan berlaku pada suatu permintaan, izin yang dihasilkan lebih rumit untuk dipahami. Untuk mempelajari cara AWS menentukan apakah akan mengizinkan permintaan saat beberapa jenis kebijakan terlibat, lihat Logika evaluasi kebijakan di Panduan Pengguna IAM.

Bagaimana AWS Proton bekerja dengan IAM

Sebelum Anda menggunakan IAM untuk mengelola akses AWS Proton, pelajari fitur IAM yang tersedia untuk digunakan. AWS Proton

Fitur IAM yang dapat Anda gunakan dengan AWS Proton

Fitur IAM	AWS Proton dukungan
Kebijakan berbasis identitas	Ya
Kebijakan berbasis sumber daya	Tidak
Tindakan kebijakan	Ya
Sumber daya kebijakan	Ya
Kunci kondisi kebijakan	Ya
ACLs	Tidak
ABAC (tanda dalam kebijakan)	Ya
Kredensial sementara	Ya
Izin principal	Ya
Peran layanan	Ya
Peran terkait layanan	Ya

Untuk mendapatkan tampilan tingkat tinggi tentang cara AWS Proton dan Layanan AWS pekerjaan lainnya dengan sebagian besar fitur IAM, lihat Layanan AWS yang berfungsi dengan IAM di Panduan Pengguna IAM.

Kebijakan berbasis identitas untuk AWS Proton

Mendukung kebijakan berbasis identitas: Ya

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.

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. Untuk mempelajari semua elemen yang dapat Anda gunakan dalam kebijakan JSON, lihat Referensi elemen kebijakan JSON IAM dalam Panduan Pengguna IAM.

Contoh kebijakan berbasis identitas untuk AWS Proton

Untuk melihat contoh kebijakan AWS Proton berbasis identitas, lihat. Contoh kebijakan berbasis identitas untuk AWS Proton

Kebijakan berbasis sumber daya dalam AWS Proton

Mendukung kebijakan berbasis sumber daya: Tidak

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 principal tertentu pada sumber daya tersebut dan dalam kondisi apa. Anda harus menentukan principal dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup akun, pengguna, peran, pengguna federasi, atau. Layanan AWS

Untuk mengaktifkan akses lintas akun, Anda dapat menentukan secara spesifik seluruh akun atau entitas IAM di akun lain sebagai principal dalam kebijakan berbasis sumber daya. Untuk informasi selengkapnya, lihat Akses sumber daya lintas akun di IAM dalam Panduan Pengguna IAM.

Tindakan kebijakan untuk AWS Proton

Mendukung tindakan kebijakan: Ya

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Yaitu, di mana utama dapat melakukan tindakan pada sumber daya, dan dalam kondisi apa.

Elemen `Action` dari kebijakan JSON menjelaskan tindakan yang dapat Anda gunakan untuk mengizinkan atau menolak akses dalam sebuah kebijakan. Sertakan tindakan dalam kebijakan untuk memberikan izin untuk melakukan operasi terkait.

Untuk melihat daftar AWS Proton tindakan, lihat Tindakan yang ditentukan oleh AWS Proton dalam Referensi Otorisasi Layanan.

Tindakan kebijakan AWS Proton menggunakan awalan berikut sebelum tindakan:

```
proton
```

Untuk menetapkan secara spesifik beberapa tindakan dalam satu pernyataan, pisahkan tindakan tersebut dengan koma.

```
"Action": [
      "proton:action1",
      "proton:action2"
         ]
```

Anda juga dapat menentukan beberapa tindakan menggunakan wildcard (*). Sebagai contoh, untuk menentukan semua tindakan yang dimulai dengan kata `List`, sertakan tindakan berikut:

```
"Action": "proton:List*"
```

Untuk melihat contoh kebijakan AWS Proton berbasis identitas, lihat. Contoh kebijakan berbasis identitas untuk AWS Proton

Sumber daya kebijakan untuk AWS Proton

Mendukung sumber daya kebijakan: Ya

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Yaitu, di mana utama dapat melakukan tindakan pada sumber daya, dan dalam kondisi apa.

Elemen kebijakan JSON `Resource` menentukan objek yang menjadi target penerapan tindakan. Praktik terbaiknya, tentukan sumber daya menggunakan Amazon Resource Name (ARN). Untuk tindakan yang tidak mendukung izin di tingkat sumber daya, gunakan wildcard (*) untuk menunjukkan bahwa pernyataan tersebut berlaku untuk semua sumber daya.

```
"Resource": "*"
```

Untuk melihat daftar jenis sumber daya dan jenis AWS Proton sumber daya ARNs, lihat Sumber daya yang ditentukan oleh AWS Proton dalam Referensi Otorisasi Layanan. Untuk mempelajari dengan tindakan mana Anda dapat menentukan ARN setiap sumber daya, lihat Tindakan yang ditentukan oleh AWS Proton.

Untuk melihat contoh kebijakan AWS Proton berbasis identitas, lihat. Contoh kebijakan berbasis identitas untuk AWS Proton

Kunci kondisi kebijakan untuk AWS Proton

Mendukung kunci kondisi kebijakan khusus layanan: Yes

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.

Elemen `Condition` menentukan ketika pernyataan dieksekusi berdasarkan kriteria yang ditetapkan. 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. Untuk melihat semua kunci kondisi AWS global, lihat kunci konteks kondisi AWS global di Panduan Pengguna IAM.

Untuk melihat daftar kunci AWS Proton kondisi, lihat Kunci kondisi untuk AWS Proton dalam Referensi Otorisasi Layanan. Untuk mempelajari tindakan dan sumber daya yang dapat Anda gunakan kunci kondisi, lihat Tindakan yang ditentukan oleh AWS Proton.

Untuk melihat contoh condition-key-based kebijakan untuk membatasi akses ke sumber daya, lihatContoh kebijakan berbasis kondisi-kunci untuk AWS Proton.

Daftar kontrol akses (ACLs) di AWS Proton

Mendukung ACLs: Tidak

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.

Daftar kontrol akses (ACLs) adalah daftar penerima hibah yang dapat Anda lampirkan ke sumber daya. Mereka memberikan izin akun untuk mengakses sumber daya tempat mereka dilampirkan.

Kontrol akses berbasis atribut (ABAC) dengan AWS Proton

Mendukung ABAC (tanda dalam kebijakan): Ya

Kontrol akses berbasis atribut (ABAC) adalah strategi otorisasi yang menentukan izin berdasarkan atribut tanda. Anda dapat melampirkan tag ke entitas dan AWS sumber daya IAM, lalu merancang kebijakan ABAC untuk mengizinkan operasi saat tag prinsipal cocok dengan tag pada sumber daya.

Untuk mengendalikan akses berdasarkan tanda, berikan informasi tentang tanda di elemen kondisi dari kebijakan menggunakan kunci kondisi `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, atau `aws:TagKeys`.

Jika sebuah layanan mendukung ketiga kunci kondisi untuk setiap jenis sumber daya, nilainya adalah Ya untuk layanan tersebut. Jika suatu layanan mendukung ketiga kunci kondisi untuk hanya beberapa jenis sumber daya, nilainya adalah Parsial.

Untuk informasi selengkapnya tentang ABAC, lihat Tentukan izin dengan otorisasi ABAC dalam Panduan Pengguna IAM. Untuk melihat tutorial yang menguraikan langkah-langkah pengaturan ABAC, lihat Menggunakan kontrol akses berbasis atribut (ABAC) dalam Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang menandai AWS Proton sumber daya, lihatAWS Proton sumber daya dan penandaan.

Menggunakan kredensi sementara dengan AWS Proton

Mendukung kredensial sementara: Ya

Kredensi sementara menyediakan akses jangka pendek ke AWS sumber daya dan secara otomatis dibuat saat Anda menggunakan federasi atau beralih peran. AWS merekomendasikan agar Anda menghasilkan kredensyal sementara secara dinamis alih-alih menggunakan kunci akses jangka panjang. Untuk informasi selengkapnya, lihat Kredensial keamanan sementara di IAM dan Layanan AWS yang berfungsi dengan IAM dalam Panduan Pengguna IAM.

Izin utama lintas layanan untuk AWS Proton

Mendukung sesi akses terusan (FAS): Ya

Sesi akses teruskan (FAS) menggunakan izin dari pemanggilan utama Layanan AWS, dikombinasikan dengan permintaan Layanan AWS untuk membuat permintaan ke layanan hilir. Untuk detail kebijakan ketika mengajukan permintaan FAS, lihat Sesi akses terusan.

Peran layanan untuk AWS Proton

Mendukung peran layanan: Ya

Peran layanan adalah peran IAM yang diambil oleh sebuah 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.

Untuk informasi selengkapnya, lihat AWS Proton Contoh kebijakan peran layanan IAM.

> **Warning**
>
> Mengubah izin untuk peran layanan dapat merusak AWS Proton fungsionalitas. Edit peran layanan hanya jika AWS Proton memberikan panduan untuk melakukannya.

Peran terkait layanan untuk AWS Proton

Mendukung peran terkait layanan: Ya

Peran terkait layanan adalah jenis peran layanan yang ditautkan ke. Layanan AWS Layanan tersebut dapat menjalankan peran untuk melakukan tindakan atas nama Anda. Peran terkait layanan muncul di Anda Akun AWS dan dimiliki oleh layanan. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.

Untuk informasi selengkapnya, lihat Menggunakan peran terkait layanan untuk AWS Proton.

Contoh kebijakan untuk AWS Proton

Temukan contoh kebijakan AWS Proton IAM di bagian berikut.

Topik

- Contoh kebijakan berbasis identitas untuk AWS Proton
- AWS Proton Contoh kebijakan peran layanan IAM
- Contoh kebijakan berbasis kondisi-kunci untuk AWS Proton

Contoh kebijakan berbasis identitas untuk AWS Proton

Secara default, pengguna dan peran tidak memiliki izin untuk membuat atau mengubah sumber daya AWS Proton . Untuk memberikan izin kepada pengguna untuk melakukan tindakan di sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM.

Untuk mempelajari cara membuat kebijakan berbasis identitas IAM dengan menggunakan contoh dokumen kebijakan JSON ini, lihat Membuat kebijakan IAM (konsol) di Panduan Pengguna IAM.

Untuk detail tentang tindakan dan jenis sumber daya yang ditentukan oleh AWS Proton, termasuk format ARNs untuk setiap jenis sumber daya, lihat Kunci tindakan, sumber daya, dan kondisi AWS Proton dalam Referensi Otorisasi Layanan.

Topik

- Praktik terbaik kebijakan
- Tautan ke contoh kebijakan berbasis Identitas untuk AWS Proton

Praktik terbaik kebijakan

Kebijakan berbasis identitas menentukan apakah seseorang dapat membuat, mengakses, atau menghapus AWS Proton sumber daya 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 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.

Tautan ke contoh kebijakan berbasis Identitas untuk AWS Proton

Tautan ke contoh contoh kebijakan berbasis identitas untuk AWS Proton

- AWS kebijakan terkelola untuk AWS Proton
- AWS Proton Contoh kebijakan peran layanan IAM
- Contoh kebijakan berbasis kondisi-kunci untuk AWS Proton

AWS Proton Contoh kebijakan peran layanan IAM

Administrator memiliki dan mengelola sumber daya yang AWS Proton dibuat seperti yang didefinisikan oleh template lingkungan dan layanan. Mereka melampirkan peran layanan IAM ke akun mereka yang memungkinkan AWS Proton untuk membuat sumber daya atas nama mereka.

Administrator menyediakan peran dan AWS Key Management Service kunci IAM untuk sumber daya yang kemudian dimiliki dan dikelola oleh pengembang saat AWS Proton menyebarkan aplikasi mereka sebagai AWS Proton layanan di lingkungan. AWS Proton Untuk informasi selengkapnya tentang AWS KMS dan enkripsi data, lihatPerlindungan data di AWS Proton.

Peran layanan adalah peran Amazon Web Services (IAM) yang memungkinkan Anda melakukan panggilan AWS Proton ke sumber daya atas nama Anda. Jika Anda menentukan peran layanan, AWS Proton menggunakan kredensial peran tersebut. Gunakan peran layanan untuk secara eksplisit menentukan tindakan yang AWS Proton dapat dilakukan.

Anda membuat peran layanan dan kebijakan izinnya dengan layanan IAM. Untuk informasi selengkapnya tentang membuat peran layanan, lihat Membuat peran untuk mendelegasikan izin ke AWS layanan di Panduan Pengguna IAM.

AWS Proton peran layanan untuk penyediaan menggunakan CloudFormation

Sebagai anggota tim platform, Anda dapat sebagai administrator membuat peran AWS Proton layanan dan menyediakannya AWS Proton saat Anda membuat lingkungan sebagai peran CloudFormation layanan lingkungan (`protonServiceRoleArn`parameter tindakan CreateEnvironmentAPI). Peran ini memungkinkan AWS Proton untuk melakukan panggilan API ke layanan lain atas nama Anda ketika lingkungan atau instans layanan apa pun yang berjalan di dalamnya menggunakan penyediaan AWS-managed dan AWS CloudFormation untuk menyediakan infrastruktur.

Kami menyarankan Anda menggunakan peran IAM berikut dan kebijakan kepercayaan untuk peran AWS Proton layanan Anda. Saat Anda menggunakan AWS Proton konsol untuk membuat lingkungan dan memilih untuk membuat peran baru, ini adalah kebijakan yang AWS Proton menambahkan peran layanan yang dibuatnya untuk Anda. Saat mencantumkan izin pada kebijakan ini, ingatlah bahwa AWS Proton gagal pada `Access Denied` kesalahan.

> **Important**
>
> Perlu diketahui bahwa kebijakan yang ditampilkan dalam contoh berikut memberikan hak administrator kepada siapa saja yang dapat mendaftarkan templat ke akun Anda. Karena kami tidak tahu sumber daya mana yang akan Anda tentukan di AWS Proton templat Anda, kebijakan ini memiliki izin yang luas. Kami menyarankan Anda untuk mencatat izin ke sumber daya tertentu yang akan digunakan di lingkungan Anda.

AWS Proton contoh kebijakan peran layanan untuk CloudFormation

Ganti *123456789012* dengan Akun AWS ID Anda.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:CancelUpdateStack",
        "cloudformation:ContinueUpdateRollback",
        "cloudformation:CreateChangeSet",
        "cloudformation:CreateStack",
        "cloudformation:DeleteChangeSet",
        "cloudformation:DeleteStack",
        "cloudformation:DescribeChangeSet",
        "cloudformation:DescribeStackDriftDetectionStatus",
        "cloudformation:DescribeStackEvents",
        "cloudformation:DescribeStackResourceDrifts",
        "cloudformation:DescribeStacks",
        "cloudformation:DetectStackResourceDrift",
        "cloudformation:ExecuteChangeSet",
        "cloudformation:ListChangeSets",
        "cloudformation:ListStackResources",
        "cloudformation:UpdateStack"
      ],
      "Resource": "arn:aws:cloudformation:*:123456789012:stack/AWSProton-*"
    },
    {
      "Effect": "Allow",
      "NotAction": [
        "organizations:*",
        "account:*"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": [
            "cloudformation.amazonaws.com"
          ]
```

```
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "organizations:DescribeOrganization",
        "account:ListRegions"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": [
            "cloudformation.amazonaws.com"
          ]
        }
      }
    }
  ]
}
```

AWS Proton kebijakan kepercayaan layanan

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Sid": "ServiceTrustRelationshipWithConfusedDeputyPrevention",
    "Effect": "Allow",
    "Principal": {
      "Service": "proton.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      },
      "ArnLike": {
        "aws:SourceArn": "arn:aws:proton:*:123456789012:environment/*"
      }
    }
```

```
  }
}
```

Kebijakan peran layanan penyediaan AWS terkelola yang dicakup

Berikut ini adalah contoh kebijakan peran AWS Proton layanan bawah cakupan yang dapat Anda gunakan jika Anda hanya memerlukan AWS Proton layanan untuk menyediakan sumber daya S3.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:CancelUpdateStack",
        "cloudformation:ContinueUpdateRollback",
        "cloudformation:CreateChangeSet",
        "cloudformation:CreateStack",
        "cloudformation:DeleteChangeSet",
        "cloudformation:DeleteStack",
        "cloudformation:DescribeChangeSet",
        "cloudformation:DescribeStackDriftDetectionStatus",
        "cloudformation:DescribeStackEvents",
        "cloudformation:DescribeStackResourceDrifts",
        "cloudformation:DescribeStacks",
        "cloudformation:DetectStackResourceDrift",
        "cloudformation:ExecuteChangeSet",
        "cloudformation:ListChangeSets",
        "cloudformation:ListStackResources",
        "cloudformation:UpdateStack"
      ],
      "Resource": "arn:aws:cloudformation:*:123456789012:stack/AWSProton-*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:*"
      ],
      "Resource": "*",
      "Condition": {
```

```
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": [
            "cloudformation.amazonaws.com"
          ]
        }
      }
    }
  ]
}
```

AWS Proton peran layanan untuk CodeBuild penyediaan

Sebagai anggota tim platform, Anda dapat sebagai administrator membuat peran AWS Proton layanan dan menyediakannya AWS Proton saat Anda membuat lingkungan sebagai peran CodeBuild layanan lingkungan (`codebuildRoleArn`parameter tindakan CreateEnvironmentAPI). Peran ini memungkinkan AWS Proton untuk melakukan panggilan API ke layanan lain atas nama Anda ketika lingkungan atau instans layanan apa pun yang berjalan di dalamnya menggunakan CodeBuild penyediaan untuk penyediaan infrastruktur.

Saat Anda menggunakan AWS Proton konsol untuk membuat lingkungan dan memilih untuk membuat peran baru, AWS Proton tambahkan kebijakan dengan hak administrator ke peran layanan yang dibuatnya untuk Anda. Saat Anda membuat izin peran dan cakupan Anda sendiri, ingatlah bahwa AWS Proton gagal pada `Access Denied` kesalahan.

> **Important**
>
> Ketahuilah bahwa kebijakan yang AWS Proton melekat pada peran yang dibuatnya untuk Anda memberikan hak administrator kepada siapa pun yang dapat mendaftarkan templat ke akun Anda. Karena kami tidak tahu sumber daya mana yang akan Anda tentukan di AWS Proton templat Anda, kebijakan ini memiliki izin yang luas. Kami menyarankan Anda untuk mencatat izin ke sumber daya tertentu yang akan digunakan di lingkungan Anda.

AWS Proton contoh kebijakan peran layanan untuk CodeBuild

Contoh berikut memberikan izin CodeBuild untuk menyediakan sumber daya menggunakan. AWS Cloud Development Kit (AWS CDK)

Ganti *123456789012* dengan Akun AWS ID Anda.

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [
   {
     "Action": [
       "logs:CreateLogStream",
       "logs:CreateLogGroup",
       "logs:PutLogEvents"
     ],
     "Resource": [
       "arn:aws:logs:us-east-1:123456789012:log-group:/aws/codebuild/AWSProton-
 Shell-*",
       "arn:aws:logs:us-east-1:123456789012:log-group:/aws/codebuild/AWSProton-
 Shell-*:*"
     ],
     "Effect": "Allow"
   },
   {
     "Action": "proton:NotifyResourceDeploymentStatusChange",
     "Resource": "arn:aws:proton:us-east-1:123456789012:*",
     "Effect": "Allow"
   },
   {
     "Action": "sts:AssumeRole",
     "Resource": [
       "arn:aws:iam::123456789012:role/cdk-*-deploy-role-*",
       "arn:aws:iam::123456789012:role/cdk-*-file-publishing-role-*"
     ],
     "Effect": "Allow"
   }
 ]
}
```

AWS Proton CodeBuild kebijakan kepercayaan

JSON

```
{
  "Version":"2012-10-17",
```

```
  "Statement": {
    "Sid": "CodeBuildTrustRelationshipWithConfusedDeputyPrevention",
    "Effect": "Allow",
    "Principal": {
      "Service": "codebuild.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      },
      "ArnLike": {
        "aws:SourceArn": "arn:aws:proton:*:123456789012:environment/*"
      }
    }
  }
}
```

AWS Proton peran layanan pipa

Untuk menyediakan pipeline layanan, AWS Proton perlu izin untuk melakukan panggilan API ke layanan lain. Peran layanan yang diperlukan mirip dengan peran layanan yang Anda berikan saat membuat lingkungan. Namun, peran untuk membuat pipeline dibagikan di antara semua layanan di AWS akun Anda, dan Anda memberikan peran ini sebagai setelan Akun di konsol, atau melalui tindakan UpdateAccountSettingsAPI.

Saat Anda menggunakan AWS Proton konsol untuk memperbarui setelan akun dan memilih untuk membuat peran baru untuk peran CloudFormation atau peran CodeBuild layanan, kebijakan yang AWS Proton ditambahkan ke peran layanan yang dibuatnya untuk Anda sama dengan kebijakan yang dijelaskan di bagian sebelumnya, AWS-peran penyediaan terkelola danCodeBuild peran penyediaan. Saat mencantumkan izin pada kebijakan ini, ingatlah bahwa AWS Proton gagal pada `Access Denied` kesalahan.

> **Important**
>
> Ketahuilah bahwa contoh kebijakan di bagian sebelumnya memberikan hak administrator kepada siapa saja yang dapat mendaftarkan templat ke akun Anda. Karena kami tidak tahu sumber daya mana yang akan Anda tentukan di AWS Proton templat Anda, kebijakan ini

memiliki izin yang luas. Kami menyarankan Anda untuk memasukkan izin ke sumber daya spesifik yang akan digunakan di saluran pipa Anda.

AWS Proton peran komponen

Sebagai anggota tim platform, Anda dapat sebagai administrator membuat peran AWS Proton layanan dan menyediakannya AWS Proton saat Anda membuat lingkungan sebagai peran CloudFormation komponen lingkungan (`componentRoleArn`parameter tindakan CreateEnvironmentAPI). Peran ini mencakup infrastruktur yang dapat disediakan oleh komponen yang didefinisikan secara langsung. Untuk informasi selengkapnya tentang komponen, lihatKomponen-komponen.

Contoh kebijakan berikut mendukung pembuatan komponen yang ditentukan secara langsung yang menyediakan bucket Amazon Simple Storage Service (Amazon S3) dan kebijakan akses terkait.

AWS Proton contoh kebijakan peran komponen

Ganti *123456789012* dengan Akun AWS ID Anda.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:CancelUpdateStack",
        "cloudformation:CreateChangeSet",
        "cloudformation:DeleteChangeSet",
        "cloudformation:DescribeStacks",
        "cloudformation:ContinueUpdateRollback",
        "cloudformation:DetectStackResourceDrift",
        "cloudformation:DescribeStackResourceDrifts",
        "cloudformation:DescribeStackEvents",
        "cloudformation:CreateStack",
        "cloudformation:DeleteStack",
        "cloudformation:UpdateStack",
        "cloudformation:DescribeChangeSet",
        "cloudformation:ExecuteChangeSet",
```

```
        "cloudformation:ListChangeSets",
        "cloudformation:ListStackResources"
      ],
      "Resource": "arn:aws:cloudformation:*:123456789012:stack/AWSProton-*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:CreateBucket",
        "s3:DeleteBucket",
        "s3:GetBucket*",
        "iam:CreatePolicy",
        "iam:DeletePolicy",
        "iam:GetPolicy",
        "iam:ListPolicyVersions",
        "iam:DeletePolicyVersion"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": "cloudformation.amazonaws.com"
        }
      }
    }
  ]
}
```

AWS Proton kebijakan kepercayaan komponen

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Sid": "ServiceTrustRelationshipWithConfusedDeputyPrevention",
    "Effect": "Allow",
    "Principal": {
      "Service": "proton.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringEquals": {
```

```
        "aws:SourceAccount": "123456789012"
      },
      "ArnLike": {
        "aws:SourceArn": "arn:aws:proton:*:123456789012:environment/*"
      }
    }
  }
}
```

Contoh kebijakan berbasis kondisi-kunci untuk AWS Proton

Contoh berikut kebijakan IAM menolak akses ke AWS Proton tindakan yang cocok dengan templat yang ditentukan dalam blok. `Condition` Perhatikan bahwa kunci kondisi ini hanya didukung oleh tindakan yang tercantum di Tindakan, sumber daya, dan kunci kondisi untuk AWS Proton. Untuk mengelola izin pada tindakan lain, seperti`DeleteEnvironmentTemplate`, Anda harus menggunakan kontrol akses tingkat sumber daya.

Contoh kebijakan yang menolak tindakan AWS Proton template pada template tertentu:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": ["proton:*"],
            "Resource": "*",
            "Condition": {
                "StringEqualsIfExists": {
                    "proton:EnvironmentTemplate":
 ["arn:aws:proton:region_id:123456789012:environment-template/my-environment-
template"]
                }
            }
        },
        {
            "Effect": "Deny",
            "Action": ["proton:*"],
            "Resource": "*",
            "Condition": {
```

```
                "StringEqualsIfExists": {
                    "proton:ServiceTemplate":
 ["arn:aws:proton:region_id:123456789012:service-template/my-service-template"]
                }
            }
        }
    ]
}
```

Dalam kebijakan contoh berikutnya, pernyataan tingkat Sumber Daya pertama menolak akses ke tindakan AWS Proton templat`ListServiceTemplates`, selain yang cocok dengan templat layanan yang tercantum di blok. `Resource` Pernyataan kedua menolak akses ke AWS Proton tindakan yang cocok dengan templat yang tercantum di `Condition` blok.

Contoh kebijakan yang menyangkal AWS Proton tindakan yang cocok dengan templat tertentu:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "proton:*"
            ],
            "Resource": "arn:aws:proton:us-east-1:123456789012:service-template/
my-service-template"
        },
        {
            "Effect": "Deny",
            "Action": [
                "proton:*"
            ],
            "Resource": "*",
            "Condition": {
                "StringEqualsIfExists": {
                    "proton:ServiceTemplate": [
                        "arn:aws:proton:us-east-1:123456789012:service-template/
my-service-template"
                    ]
```

```
                }
            }
        }
    ]
}
```

Contoh kebijakan akhir memungkinkan AWS Proton tindakan developer yang cocok dengan templat layanan tertentu yang tercantum dalam `Condition` blok.

Contoh kebijakan untuk mengizinkan tindakan AWS Proton pengembang yang cocok dengan templat tertentu:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "proton:ListServiceTemplates",
                "proton:ListServiceTemplateVersions",
                "proton:ListServices",
                "proton:ListServiceInstances",
                "proton:ListEnvironments",
                "proton:GetServiceTemplate",
                "proton:GetServiceTemplateVersion",
                "proton:GetService",
                "proton:GetServiceInstance",
                "proton:GetEnvironment",
                "proton:CreateService",
                "proton:UpdateService",
                "proton:UpdateServiceInstance",
                "proton:UpdateServicePipeline",
                "proton:DeleteService",
                "codestar-connections:ListConnections"
            ],
            "Resource": "*",
            "Condition": {
                "StringEqualsIfExists": {
```

```
                    "proton:ServiceTemplate":
 "arn:aws:proton:region_id:123456789012:service-template/my-service-template"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "codestar-connections:PassConnection"
            ],
            "Resource": "arn:aws:codestar-connections:*:*:connection/*",
            "Condition": {
                "StringEquals": {
                    "codestar-connections:PassedToService":
 "proton.amazonaws.com"
                }
            }
        }

    ]
}
```

AWS kebijakan terkelola untuk AWS Proton

Untuk menambahkan izin ke pengguna, grup, dan peran, lebih mudah menggunakan kebijakan AWS terkelola daripada menulis kebijakan sendiri. Dibutuhkan waktu dan keahlian untuk membuat kebijakan yang dikelola pelanggan IAM yang hanya memberi tim Anda izin yang mereka butuhkan. Untuk memulai dengan cepat, Anda dapat menggunakan kebijakan AWS terkelola kami. Kebijakan ini mencakup kasus penggunaan umum dan tersedia di Akun AWS Anda. Untuk informasi selengkapnya tentang kebijakan AWS AWS terkelola, lihat kebijakan terkelola di Panduan Pengguna IAM.

Layanan AWS memelihara dan memperbarui kebijakan AWS terkelola. Anda tidak dapat mengubah izin dalam kebijakan AWS terkelola. Layanan terkadang menambahkan izin tambahan ke kebijakan yang dikelola AWS untuk mendukung fitur-fitur baru. Jenis pembaruan ini akan memengaruhi semua identitas (pengguna, grup, dan peran) di mana kebijakan tersebut dilampirkan. Layanan kemungkinan besar akan memperbarui kebijakan yang dikelola AWS saat ada fitur baru yang diluncurkan atau saat ada operasi baru yang tersedia. Layanan tidak menghapus izin dari kebijakan AWS terkelola, sehingga pembaruan kebijakan tidak akan merusak izin yang ada.

Selain itu, AWS mendukung kebijakan terkelola untuk fungsi pekerjaan yang mencakup beberapa layanan. Misalnya, kebijakan ReadOnlyAccess AWS terkelola menyediakan akses hanya-baca ke semua Layanan AWS dan sumber daya. Saat layanan meluncurkan fitur baru, AWS menambahkan izin hanya-baca untuk operasi dan sumber daya baru. Untuk melihat daftar dan deskripsi dari kebijakan fungsi tugas, lihat kebijakan yang dikelola AWS untuk fungsi tugas di Panduan Pengguna IAM.

AWS Proton menyediakan kebijakan IAM terkelola dan hubungan kepercayaan yang dapat Anda lampirkan ke pengguna, grup, atau peran yang memungkinkan berbagai tingkat kontrol atas sumber daya dan operasi API. Anda dapat menerapkan kebijakan ini secara langsung atau menggunakannya sebagai titik awal untuk membuat kebijakan Anda sendiri.

Hubungan kepercayaan berikut digunakan untuk setiap kebijakan yang AWS Proton dikelola.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Sid": "ExampleTrustRelationshipWithProtonConfusedDeputyPrevention",
    "Effect": "Allow",
    "Principal": {
      "Service": "proton.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      },
      "ArnLike": {
        "aws:SourceArn": "arn:aws:proton:*:123456789012:environment/*"
      }
    }
  }
}
```

AWS kebijakan terkelola: AWSProton FullAccess

Anda dapat melampirkan `AWSProtonFullAccess` ke entitas IAM Anda. AWS Proton juga melampirkan kebijakan ini ke peran layanan yang memungkinkan AWS Proton untuk melakukan tindakan atas nama Anda.

Kebijakan ini memberikan izin administratif yang memungkinkan akses penuh ke AWS Proton tindakan dan akses terbatas ke tindakan AWS layanan lain yang AWS Proton bergantung pada.

Kebijakan ini mencakup ruang nama tindakan kunci berikut:

- `proton`— Memungkinkan administrator akses penuh ke AWS Proton APIs.
- `iam`— Memungkinkan administrator untuk meneruskan peran ke AWS Proton. Hal ini diperlukan agar AWS Proton dapat melakukan panggilan API ke layanan lain atas nama administrator.
- `kms`— Memungkinkan administrator untuk menambahkan hibah ke kunci yang dikelola pelanggan.
- `codeconnections`— Memungkinkan administrator untuk membuat daftar dan meneruskan codeconnections sehingga mereka dapat digunakan oleh. AWS Proton

Untuk informasi selengkapnya, lihat AWSProtonFullAccess.

AWS kebijakan terkelola: AWSProton DeveloperAccess

Anda dapat melampirkan `AWSProtonDeveloperAccess` ke entitas IAM Anda. AWS Proton juga melampirkan kebijakan ini ke peran layanan yang memungkinkan AWS Proton untuk melakukan tindakan atas nama Anda.

Kebijakan ini memberikan izin yang memungkinkan akses terbatas ke AWS Proton tindakan dan AWS tindakan lain yang AWS Proton bergantung padanya. Ruang lingkup izin ini dirancang untuk mendukung peran pengembang yang membuat dan menyebarkan layanan AWS Proton .

Kebijakan ini tidak menyediakan akses ke pembuatan, penghapusan, dan pembaruan AWS Proton templat dan lingkungan APIs. Jika pengembang memerlukan izin yang lebih terbatas daripada yang disediakan kebijakan ini, sebaiknya buat kebijakan khusus yang dicakup untuk memberikan hak istimewa paling sedikit.

Kebijakan ini mencakup ruang nama tindakan kunci berikut:

- `proton`— Memungkinkan akses kontributor ke set terbatas. AWS Proton APIs

- `codeconnections`— Memungkinkan kontributor untuk daftar dan meneruskan codeconnections sehingga mereka dapat digunakan oleh. AWS Proton

Untuk informasi selengkapnya, lihat AWSProtonDeveloperAccess.

AWS kebijakan terkelola: AWSProton ReadOnlyAccess

Anda dapat melampirkan `AWSProtonReadOnlyAccess` ke entitas IAM Anda. AWS Proton juga melampirkan kebijakan ini ke peran layanan yang memungkinkan AWS Proton untuk melakukan tindakan atas nama Anda.

Kebijakan ini memberikan izin yang memungkinkan akses hanya-baca ke AWS Proton tindakan dan akses hanya-baca terbatas ke tindakan layanan lain AWS yang bergantung padanya. AWS Proton

Kebijakan ini mencakup ruang nama tindakan kunci berikut:

- `proton`— Memungkinkan kontributor akses hanya-baca ke. AWS Proton APIs

Untuk informasi selengkapnya, lihat AWSProtonReadOnlyAccess.

AWS kebijakan terkelola: AWSProton SyncServiceRolePolicy

AWS Proton melampirkan kebijakan ini ke peran AWSServiceRoleForProtonSyncterkait layanan yang memungkinkan AWS Proton untuk melakukan sinkronisasi templat.

Kebijakan ini memberikan izin yang memungkinkan akses terbatas ke AWS Proton tindakan dan tindakan AWS layanan lain yang AWS Proton bergantung padanya.

Kebijakan ini mencakup ruang nama tindakan kunci berikut:

- `proton`— Memungkinkan AWS Proton sinkronisasi akses terbatas ke AWS Proton APIs.
- `codeconnections`— Memungkinkan AWS Proton sinkronisasi akses terbatas ke CodeConnections APIs.

Untuk informasi selengkapnya, lihat AWSProtonSyncServiceRolePolicy.

AWS kebijakan terkelola: AWSProton CodeBuildProvisioningBasicAccess

Izin CodeBuild perlu menjalankan build untuk AWS Proton CodeBuild Penyediaan. Anda dapat melampirkan `AWSProtonCodeBuildProvisioningBasicAccess` ke CodeBuild Peran Penyediaan Anda.

Kebijakan ini memberikan izin minimum agar AWS Proton CodeBuild Penyediaan berfungsi. Ini memberikan izin yang memungkinkan CodeBuild untuk menghasilkan log build. Ini juga memberikan izin kepada Proton untuk membuat output Infrastructure as Code (IAc) tersedia bagi pengguna. AWS Proton Itu tidak memberikan izin yang dibutuhkan oleh alat IAc untuk mengelola infrastruktur.

Kebijakan ini mencakup ruang nama tindakan kunci berikut:

- `logs`- Memungkinkan CodeBuild untuk menghasilkan log build. Tanpa izin ini, CodeBuild akan gagal untuk memulai.
- `proton`- Memungkinkan perintah CodeBuild Provisioning untuk memanggil `aws proton notify-resource-deployment-status-change` untuk memperbarui output IAAC untuk sumber daya tertentu. AWS Proton

Untuk informasi selengkapnya, lihat AWSProtonCodeBuildProvisioningBasicAccess.

AWS kebijakan terkelola: AWSProton CodeBuildProvisioningServiceRolePolicy

AWS Proton melampirkan kebijakan ini ke peran AWSServiceRoleForProtonCodeBuildProvisioningterkait layanan yang memungkinkan AWS Proton untuk melakukan CodeBuild penyediaan berbasis.

Kebijakan ini memberikan izin yang memungkinkan akses terbatas ke tindakan AWS layanan yang AWS Proton bergantung pada.

Kebijakan ini mencakup ruang nama tindakan kunci berikut:

- `cloudformation`— Memungkinkan penyediaan AWS Proton CodeBuild berbasis akses terbatas ke. CloudFormation APIs
- `codebuild`— Memungkinkan penyediaan AWS Proton CodeBuild berbasis akses terbatas ke. CodeBuild APIs
- `iam`— Memungkinkan administrator untuk meneruskan peran ke AWS Proton. Hal ini diperlukan agar AWS Proton dapat melakukan panggilan API ke layanan lain atas nama administrator.

- `servicequotas`— Memungkinkan AWS Proton untuk memeriksa batas build CodeBuild bersamaan, yang memastikan antrian build yang tepat.

Untuk informasi selengkapnya, lihat AWSProtonCodeBuildProvisioningServiceRolePolicy.

AWS kebijakan terkelola: AWSProton ServiceGitSyncServiceRolePolicy

AWS Proton melampirkan kebijakan ini ke peran AWSServiceRoleForProtonServiceSyncterkait layanan yang memungkinkan AWS Proton untuk melakukan sinkronisasi layanan.

Kebijakan ini memberikan izin yang memungkinkan akses terbatas ke AWS Proton tindakan dan tindakan AWS layanan lain yang AWS Proton bergantung padanya.

Kebijakan ini mencakup ruang nama tindakan kunci berikut:

- `proton`— Memungkinkan AWS Proton sinkronisasi akses terbatas ke AWS Proton APIs.

Untuk informasi selengkapnya, lihat AWSProtonServiceGitSyncServiceRolePolicy.

AWS Proton pembaruan kebijakan AWS terkelola

Lihat detail tentang pembaruan kebijakan AWS terkelola AWS Proton sejak layanan ini mulai melacak perubahan ini. Untuk peringatan otomatis tentang perubahan pada halaman ini, berlangganan umpan RSS di halaman Riwayat AWS Proton dokumen.

Ubah	Deskripsi	Tanggal
AWSProtonCodeBuild ProvisioningServiceRolePolicy — Permbaruan ke kebijakan yang sudah ada	Kebijakan terkelola untuk peran terkait layanan yang memungkinkan AWS Proton untuk melakukan penyediaan CodeBuild berbasis sekarang memberikan izin untuk memanggil tindakan dan API. CloudFormation `TagResource` `UntagResource` Izin ini diperlukan untuk melakukan	Juni 15, 2024

Ubah	Deskripsi	Tanggal
	operasi penandaan pada sumber daya.	
AWSProtonFullAccess – Pembaruan ke kebijakan yang ada	Kebijakan terkelola untuk peran terkait layanan untuk menggunakan sinkronisasi Git dengan repositori Git telah diperbarui untuk sumber daya dengan kedua awalan layanan. Untuk informasi selengkapnya, lihat Menggunakan peran terkait layanan untuk kebijakan AWS CodeConnections dan Terkelola.	April 25, 2024
AWSProtonDeveloperAccess – Pembaruan ke kebijakan yang ada	Kebijakan terkelola untuk peran terkait layanan untuk menggunakan sinkronisasi Git dengan repositori Git telah diperbarui untuk sumber daya dengan kedua awalan layanan. Untuk informasi selengkapnya, lihat Menggunakan peran terkait layanan untuk kebijakan AWS CodeConnections dan Terkelola.	April 25, 2024

Ubah	Deskripsi	Tanggal
AWSProtonSyncServiceRolePolicy – Pembaruan ke kebijakan yang ada	Kebijakan terkelola untuk peran terkait layanan untuk menggunakan sinkronisasi Git dengan repositori Git telah diperbarui untuk sumber daya dengan kedua awalan layanan. Untuk informasi selengkapnya, lihat Menggunakan peran terkait layanan untuk kebijakan AWS CodeConnections dan Terkelola.	April 25, 2024
AWSProtonCodeBuildProvisioningServiceRolePolicy – Pembaruan ke kebijakan yang ada	AWS Proton memperbarui kebijakan ini untuk menambahkan izin guna memastikan akun memiliki batas build CodeBuild bersamaan yang diperlukan untuk menggunakan CodeBuild Penyediaan.	12 Mei 2023
AWSProtonServiceGitSyncServiceRolePolicy – Kebijakan baru	AWS Proton menambahkan kebijakan baru untuk memungkinkan AWS Proton melakukan sinkronisasi layanan. Kebijakan ini digunakan dalam peran AWSServiceRoleForProtonServiceSyncterkait layanan.	31 Maret 2023

Ubah	Deskripsi	Tanggal
AWSProtonDeveloperAccess – Pembaruan ke kebijakan yang ada	AWS Proton menambahkan `GetResourcesSummary` tindakan baru yang memungkinkan Anda melihat ringkasan templat, sumber daya templat yang digunakan, dan sumber daya yang kedaluwarsa.	18 November 2022
AWSProtonReadOnlyAccess – Pembaruan ke kebijakan yang ada	AWS Proton menambahkan `GetResourcesSummary` tindakan baru yang memungkinkan Anda melihat ringkasan templat, sumber daya templat yang digunakan, dan sumber daya yang kedaluwarsa.	18 November 2022
AWSProtonCodeBuildProvisioningBasicAccess – Kebijakan baru	AWS Proton menambahkan kebijakan baru yang memberikan izin CodeBuild yang diperlukan untuk menjalankan build untuk AWS Proton CodeBuild Penyediaan.	16 November 2022

Ubah	Deskripsi	Tanggal
AWSProtonSyncServiceRolePolicy – Kebijakan baru	AWS Proton menambahkan kebijakan baru untuk memungkinkan AWS Proton melakukan operasi yang terkait dengan penyediaan CodeBuild berbasis. Kebijakan ini digunakan dalam peran AWSServiceRoleForProtonCodeBuildProvisioningterkait layanan.	September 02, 2022
AWSProtonFullAccess – Pembaruan ke kebijakan yang ada	AWS Proton memperbarui kebijakan ini untuk menyediakan akses ke operasi AWS Proton API baru dan untuk memperbaiki masalah izin untuk beberapa operasi AWS Proton konsol.	Maret 30, 2022
AWSProtonDeveloperAccess – Pembaruan ke kebijakan yang ada	AWS Proton perbarui kebijakan ini untuk menyediakan akses ke operasi AWS Proton API baru dan untuk memperbaiki masalah izin untuk beberapa operasi AWS Proton konsol.	Maret 30, 2022
AWSProtonReadOnlyAccess – Pembaruan ke kebijakan yang ada	AWS Proton perbarui kebijakan ini untuk menyediakan akses ke operasi AWS Proton API baru dan untuk memperbaiki masalah izin untuk beberapa operasi AWS Proton konsol.	Maret 30, 2022

Ubah	Deskripsi	Tanggal
AWSProtonSyncServiceRolePolicy – Kebijakan baru	AWS Proton menambahkan kebijakan baru untuk memungkinkan AWS Proton melakukan operasi yang terkait dengan sinkronisasi templat. Kebijakan ini digunakan dalam peran AWSServiceRoleForProtonSyncterkait layanan.	23 November 2021
AWSProtonFullAccess – Kebijakan baru	AWS Proton menambahkan kebijakan baru untuk menyediakan akses peran administratif ke operasi AWS Proton API dan ke AWS Proton konsol.	Juni 09, 2021
AWSProtonDeveloperAccess – Kebijakan baru	AWS Proton menambahkan kebijakan baru untuk menyediakan akses peran pengembang ke operasi AWS Proton API dan ke AWS Proton konsol.	Juni 09, 2021
AWSProtonReadOnlyAccess – Kebijakan baru	AWS Proton menambahkan kebijakan baru untuk menyediakan akses hanya-baca ke operasi AWS Proton API dan ke konsol. AWS Proton	Juni 09, 2021
AWS Proton mulai melacak perubahan.	AWS Proton mulai melacak perubahan untuk kebijakan yang AWS dikelola.	Juni 09, 2021

Menggunakan peran terkait layanan untuk AWS Proton

AWS Proton menggunakan AWS Identity and Access Management peran terkait layanan (IAM). Peran terkait layanan adalah jenis unik peran IAM yang ditautkan langsung ke. AWS Proton Peran terkait layanan telah ditentukan sebelumnya oleh AWS Proton dan mencakup semua izin yang diperlukan layanan untuk memanggil AWS layanan lain atas nama Anda.

Topik

- Menggunakan peran untuk AWS Proton sinkronisasi
- Menggunakan peran untuk penyediaan CodeBuild berbasis

Menggunakan peran untuk AWS Proton sinkronisasi

AWS Proton menggunakan AWS Identity and Access Management peran terkait layanan (IAM). Peran terkait layanan adalah jenis unik peran IAM yang ditautkan langsung ke. AWS Proton Peran terkait layanan telah ditentukan sebelumnya oleh AWS Proton dan mencakup semua izin yang diperlukan layanan untuk memanggil AWS layanan lain atas nama Anda.

Peran terkait layanan membuat pengaturan AWS Proton lebih mudah karena Anda tidak perlu menambahkan izin yang diperlukan secara manual. AWS Proton mendefinisikan izin peran terkait layanan, dan kecuali ditentukan lain, hanya AWS Proton 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 AWS Proton sumber daya Anda karena Anda tidak dapat secara tidak sengaja menghapus izin untuk mengakses sumber daya.

Untuk informasi tentang layanan lain yang mendukung peran terkait layanan, silakan lihat layanan AWS yang bisa digunakan dengan IAM dan carilah layanan yang memiliki opsi Ya di kolom Peran terkait layanan. Pilih Ya dengan sebuah tautan untuk melihat dokumentasi peran terkait layanan untuk layanan tersebut.

Izin peran terkait layanan untuk AWS Proton

AWS Proton menggunakan dua peran terkait layanan bernama AWSServiceRoleForProtonSyncdan. AWSServiceRoleForProtonServiceSync

Peran tertaut layanan AWSServiceRoleForProtonSync memercayai layanan berikut untuk mengambil peran tersebut:

- `sync.proton.amazonaws.com`

Kebijakan izin peran bernama `AWSProtonSyncServiceRolePolicy` memungkinkan AWS Proton untuk menyelesaikan tindakan berikut pada sumber daya yang ditentukan:

- Tindakan: membuat, mengelola, dan membaca pada AWS Proton template dan versi template
- Tindakan: gunakan koneksi pada CodeConnections

Untuk informasi selengkapnya tentang kebijakan ini, lihat AWS kebijakan terkelola: AWSProton SyncServiceRolePolicy.

Peran tertaut layanan AWSServiceRoleForProtonServiceSync memercayai layanan berikut untuk mengambil peran tersebut:

- `service-sync.proton.amazonaws.com`

Kebijakan izin peran bernama `AWSProtonServiceGitSyncServiceRolePolicy` memungkinkan AWS Proton untuk menyelesaikan tindakan berikut pada sumber daya yang ditentukan:

- Tindakan: membuat, mengelola, dan membaca AWS Proton layanan dan instance layanan

Untuk informasi selengkapnya tentang kebijakan ini, lihat AWS kebijakan terkelola: AWSProton ServiceGitSyncServiceRolePolicy.

Anda harus mengonfigurasi izin untuk mengizinkan entitas IAM (seperti pengguna, grup, atau peran) untuk membuat, mengedit, atau menghapus peran terkait layanan. Untuk informasi selengkapnya, lihat Izin peran tertaut layanan dalam Panduan Pengguna IAM.

Membuat peran terkait layanan untuk AWS Proton

Anda tidak perlu membuat peran terkait layanan secara manual. Saat Anda mengonfigurasi repositori atau layanan untuk sinkronisasi AWS Proton di Konsol Manajemen AWS, API AWS CLI, atau AWS API, AWS Proton buat peran terkait layanan untuk Anda.

Jika Anda menghapus peran terkait layanan ini, dan ingin membuatnya lagi, Anda dapat mengulangi proses yang sama untuk membuat kembali peran tersebut di akun Anda. Saat Anda mengonfigurasi

repositori atau layanan untuk sinkronisasi AWS Proton, AWS Proton buat peran terkait layanan untuk Anda lagi.

Untuk membuat ulang peran AWSServiceRoleForProtonSyncterkait layanan,
Anda ingin mengonfigurasi repositori untuk sinkronisasi, dan untuk membuat ulang AWSServiceRoleForProtonServiceSync, Anda ingin mengonfigurasi layanan untuk sinkronisasi.

Mengedit peran terkait layanan untuk AWS Proton

AWS Proton tidak memungkinkan Anda untuk mengedit peran AWSServiceRoleForProtonSyncterkait layanan. Setelah membuat peran terkait layanan, Anda tidak dapat mengubah nama peran karena berbagai entitas mungkin mereferensikan peran tersebut. Namun, Anda dapat menyunting penjelasan peran menggunakan IAM. Untuk informasi selengkapnya, lihat Mengedit peran terkait layanan dalam Panduan Pengguna IAM.

Menghapus peran terkait layanan untuk AWS Proton

Anda tidak perlu menghapus peran AWSServiceRoleForProtonSync secara manual. Saat Anda menghapus semua repositori AWS Proton tertaut untuk sinkronisasi repositori di Konsol Manajemen AWS, API AWS CLI, atau AWS API, AWS Proton membersihkan sumber daya dan menghapus peran terkait layanan untuk Anda.

Wilayah yang didukung untuk AWS Proton peran terkait layanan

AWS Proton mendukung penggunaan peran terkait layanan di semua Wilayah AWS tempat layanan tersedia. Untuk informasi selengkapnya, lihat AWS Proton titik akhir dan kuota di. Referensi Umum AWS

Menggunakan peran untuk penyediaan CodeBuild berbasis

AWS Proton menggunakan AWS Identity and Access Management peran terkait layanan (IAM). Peran terkait layanan adalah jenis unik peran IAM yang ditautkan langsung ke. AWS Proton Peran terkait layanan telah ditentukan sebelumnya oleh AWS Proton dan mencakup semua izin yang diperlukan layanan untuk memanggil AWS layanan lain atas nama Anda.

Peran terkait layanan membuat pengaturan AWS Proton lebih mudah karena Anda tidak perlu menambahkan izin yang diperlukan secara manual. AWS Proton mendefinisikan izin peran terkait layanan, dan kecuali ditentukan lain, hanya AWS Proton 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 AWS Proton sumber daya Anda karena Anda tidak dapat secara tidak sengaja menghapus izin untuk mengakses sumber daya.

Untuk informasi tentang layanan lain yang mendukung peran terkait layanan, silakan lihat layanan AWS yang bisa digunakan dengan IAM dan carilah layanan yang memiliki opsi Ya di kolom Peran terkait layanan. Pilih Ya dengan sebuah tautan untuk melihat dokumentasi peran terkait layanan untuk layanan tersebut.

Izin peran terkait layanan untuk AWS Proton

AWS Proton menggunakan peran terkait layanan bernama AWSServiceRoleForProtonCodeBuildProvisioning— Peran Tertaut Layanan untuk AWS Proton CodeBuild penyediaan.

Peran tertaut layanan AWSServiceRoleForProtonCodeBuildProvisioning memercayai layanan berikut untuk mengambil peran tersebut:

- `codebuild.proton.amazonaws.com`

Kebijakan izin peran bernama `AWSProtonCodeBuildProvisioningServiceRolePolicy` memungkinkan AWS Proton untuk menyelesaikan tindakan berikut pada sumber daya yang ditentukan:

- Tindakan: membuat, mengelola, dan membaca CloudFormation tumpukan dan transformasi
- Tindakan: membuat, mengelola, dan membaca CodeBuild proyek dan membangun

Untuk informasi selengkapnya tentang kebijakan ini, lihat AWS kebijakan terkelola: AWSProton CodeBuildProvisioningServiceRolePolicy.

Anda harus mengonfigurasi izin untuk mengizinkan entitas IAM (seperti pengguna, grup, atau peran) untuk membuat, mengedit, atau menghapus peran terkait layanan. Untuk informasi selengkapnya, lihat Izin peran tertaut layanan dalam Panduan Pengguna IAM.

Membuat peran terkait layanan untuk AWS Proton

Anda tidak perlu membuat peran terkait layanan secara manual. Saat Anda membuat lingkungan yang menggunakan penyediaan CodeBuild berbasis di AWS Proton dalam Konsol Manajemen AWS, the, atau AWS API AWS CLI, AWS Proton buat peran terkait layanan untuk Anda.

Jika Anda menghapus peran terkait layanan ini, dan ingin membuatnya lagi, Anda dapat mengulangi proses yang sama untuk membuat kembali peran tersebut di akun Anda. Saat Anda membuat lingkungan yang menggunakan penyediaan CodeBuild berbasis di AWS Proton, AWS Proton buat peran terkait layanan untuk Anda lagi.

Mengedit peran terkait layanan untuk AWS Proton

AWS Proton tidak memungkinkan Anda untuk mengedit peran AWSServiceRoleForProtonCodeBuildProvisioningterkait layanan. Setelah membuat peran terkait layanan, Anda tidak dapat mengubah nama peran karena berbagai entitas mungkin merujuk peran tersebut. Namun, Anda dapat mengedit penjelasan peran menggunakan IAM. Untuk informasi selengkapnya, lihat Mengedit peran terkait layanan dalam Panduan Pengguna IAM.

Menghapus peran terkait layanan untuk AWS Proton

Jika Anda tidak perlu lagi menggunakan fitur atau layanan yang memerlukan peran terkait layanan, sebaiknya hapus peran tersebut. Dengan begitu, Anda tidak perlu lagi memantau atau memelihara entitas yang tidak digunakan. Namun, Anda harus menghapus semua lingkungan dan layanan (instance dan pipeline) yang menggunakan penyediaan CodeBuild berbasis AWS Proton sebelum Anda dapat menghapusnya secara manual.

Menghapus peran tertaut layanan secara manual

Gunakan konsol IAM, the AWS CLI, atau AWS API untuk menghapus peran AWSServiceRoleForProtonCodeBuildProvisioningterkait layanan. Untuk informasi selengkapnya, lihat Menghapus peran terkait layanan dalam Panduan Pengguna IAM.

Wilayah yang didukung untuk AWS Proton peran terkait layanan

AWS Proton mendukung penggunaan peran terkait layanan di semua Wilayah AWS tempat layanan tersedia. Untuk informasi selengkapnya, lihat AWS Proton titik akhir dan kuota di. Referensi Umum AWS

Memecahkan masalah AWS Proton identitas dan akses

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang mungkin Anda temui saat bekerja dengan AWS Proton dan IAM.

Topik

- Saya tidak berwenang untuk melakukan tindakan di AWS Proton

- Saya tidak berwenang untuk melakukan iam: PassRole
- Saya ingin mengizinkan orang di luar saya Akun AWS untuk mengakses AWS Proton sumber daya saya

Saya tidak berwenang untuk melakukan tindakan di AWS Proton

Jika Konsol Manajemen AWS memberitahu Anda bahwa Anda tidak berwenang untuk melakukan tindakan, maka Anda harus menghubungi administrator Anda untuk bantuan. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Contoh kesalahan berikut terjadi ketika pengguna IAM `mateojackson` mencoba menggunakan konsol untuk melihat detail tentang suatu sumber daya fiktif *`my-example-widget`*, tetapi tidak memiliki izin fiktif `proton:`*`GetWidget`*.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 proton:GetWidget on resource: my-example-widget
```

Dalam hal ini, Mateo meminta administratornya untuk memperbarui kebijakannya agar dia dapat mengakses *`my-example-widget`* menggunakan `proton:`*`GetWidget`* tindakan.

Saya tidak berwenang untuk melakukan iam: PassRole

Jika Anda menerima kesalahan yang tidak diizinkan untuk melakukan `iam:PassRole` tindakan, kebijakan Anda harus diperbarui agar Anda dapat meneruskan peran AWS Proton.

Beberapa Layanan AWS memungkinkan Anda untuk meneruskan peran yang ada ke layanan tersebut alih-alih membuat peran layanan baru atau peran terkait layanan. Untuk melakukannya, Anda harus memiliki izin untuk meneruskan peran ke layanan.

Contoh kesalahan berikut terjadi ketika pengguna IAM bernama `marymajor` mencoba menggunakan konsol tersebut untuk melakukan tindakan di AWS Proton. Namun, tindakan tersebut memerlukan layanan untuk mendapatkan izin yang diberikan oleh peran layanan. Mary tidak memiliki izin untuk meneruskan peran tersebut pada layanan.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Dalam kasus ini, kebijakan Mary harus diperbarui agar dia mendapatkan izin untuk melakukan tindakan `iam:PassRole` tersebut.

Jika Anda memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Saya ingin mengizinkan orang di luar saya Akun AWS untuk mengakses AWS Proton sumber daya 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 mempelajari apakah AWS Proton mendukung fitur-fitur ini, lihatBagaimana AWS Proton 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.

Analisis konfigurasi dan kerentanan di AWS Proton

AWS Proton tidak menyediakan tambalan atau pembaruan untuk kode yang disediakan pelanggan. Pelanggan bertanggung jawab untuk memperbarui dan menerapkan tambalan ke kode mereka sendiri, termasuk kode sumber untuk layanan dan aplikasi mereka yang sedang berjalan AWS Proton dan kode yang disediakan dalam paket template layanan dan lingkungan mereka.

Pelanggan bertanggung jawab untuk memperbarui dan menambal sumber daya infrastruktur di lingkungan dan layanan mereka. AWS Proton tidak akan secara otomatis memperbarui atau

menambal sumber daya apa pun. Pelanggan harus berkonsultasi dengan dokumentasi untuk sumber daya dalam arsitektur mereka untuk memahami kebijakan penambalan masing-masing.

Selain menyediakan pembaruan lingkungan dan layanan yang diminta pelanggan ke versi minor templat layanan dan lingkungan, AWS Proton tidak menyediakan tambalan atau pembaruan pada sumber daya yang ditentukan pelanggan dalam templat layanan dan lingkungan serta bundel templat mereka.

Untuk detail selengkapnya, lihat sumber daya berikut:

- Model Tanggung Jawab Bersama
- Amazon Web Services: Gambaran Umum Proses Keamanan

Perlindungan data di AWS Proton

AWS Proton sesuai dengan model tanggung jawab AWS bersama model tanggung yang mencakup peraturan dan pedoman untuk perlindungan data. AWS bertanggung jawab untuk melindungi infrastruktur global yang menjalankan semua Layanan AWS. AWS mempertahankan kontrol atas data yang dihosting pada infrastruktur ini, termasuk kontrol konfigurasi keamanan untuk menangani konten pelanggan dan data pribadi. AWS pelanggan dan mitra APN, yang bertindak sebagai pengontrol data atau pengolah data, bertanggung jawab atas data pribadi apa pun yang mereka masukkan ke dalam AWS Cloud

Untuk tujuan perlindungan data, kami menyarankan Anda melindungi Akun AWS kredensyal dan mengatur akun pengguna individu dengan AWS Identity and Access Management (IAM), sehingga setiap pengguna hanya diberikan izin yang diperlukan untuk memenuhi tugas pekerjaan mereka. 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 AWS sumber daya. Kami merekomendasikan TLS 1.2 atau versi yang lebih baru.
- Siapkan API dan logging aktivitas pengguna dengan AWS CloudTrail.
- Gunakan solusi AWS enkripsi, bersama dengan semua kontrol keamanan default di dalamnya Layanan AWS.

Kami sangat menyarankan agar Anda tidak pernah memasukkan informasi identifikasi sensitif, seperti nomor akun pelanggan Anda, ke dalam bidang teks bentuk bebas seperti bidang Nama. Ini termasuk

saat Anda bekerja dengan AWS Proton atau lainnya Layanan AWS menggunakan konsol, API AWS CLI, atau AWS SDKs. Data apa pun yang Anda masukkan ke dalam bidang teks formulir gratis untuk pengidentifikasi sumber daya atau item serupa yang terkait dengan pengelolaan AWS sumber daya mungkin diambil untuk dimasukkan dalam log diagnostik. Saat Anda memberikan URL ke server eksternal, jangan menyertakan informasi kredensial di URL untuk memvalidasi permintaan Anda ke server tersebut.

Untuk informasi selengkapnya tentang perlindungan data, lihat postingan blog Model Tanggung Jawab Bersama AWS dan GDPR di Blog Keamanan AWS .

Enkripsi sisi server saat istirahat

Jika Anda memilih untuk mengenkripsi data sensitif dalam bundel template Anda saat istirahat di bucket S3 tempat Anda menyimpan bundel template Anda, Anda harus menggunakan kunci SSE-S3 atau SSE-KMS untuk memungkinkan untuk mengambil bundel template sehingga mereka dapat AWS Proton dilampirkan ke template terdaftar. AWS Proton

Enkripsi bergerak

Semua komunikasi layanan ke layanan dienkripsi dalam perjalanan menggunakan SSL/TLS.

AWS Proton manajemen kunci enkripsi

Di dalam AWS Proton, semua data pelanggan dienkripsi secara default menggunakan kunci yang AWS Proton dimiliki. Jika Anda menyediakan AWS KMS kunci yang dimiliki dan dikelola pelanggan, semua data pelanggan dienkripsi menggunakan kunci yang disediakan pelanggan seperti yang dijelaskan dalam paragraf berikut.

Ketika Anda membuat AWS Proton template, Anda menentukan kunci Anda dan AWS Proton menggunakan kredensi Anda untuk membuat hibah yang memungkinkan AWS Proton untuk menggunakan kunci Anda.

Jika Anda menghentikan hibah secara manual atau, menonaktifkan atau menghapus kunci yang Anda tentukan, maka AWS Proton tidak dapat membaca data yang dienkripsi oleh kunci dan melempar yang ditentukan. `ValidationException`

AWS Proton konteks enkripsi

AWS Proton mendukung header konteks enkripsi. Konteks enkripsi adalah set opsional pasangan nilai kunci yang dapat berisi informasi kontekstual tambahan tentang data. Untuk informasi lebih

lanjut tentang konteks enkripsi, lihat Konsep AWS Key Management Service - Konteks Enkripsi dalam Panduan Developer AWS Key Management Service .

Konteks enkripsi adalah sekumpulan pasangan kunci-nilai yang berisi data non-rahasia arbitrer. Saat menyertakan konteks enkripsi dalam permintaan untuk mengenkripsi data, secara AWS KMS kriptografis mengikat konteks enkripsi ke data terenkripsi. Untuk mendekripsi data, Anda harus meneruskan konteks enkripsi yang sama.

Pelanggan dapat menggunakan konteks enkripsi untuk mengidentifikasi penggunaan kunci yang dikelola pelanggan mereka dalam catatan audit dan log. Itu juga muncul dalam plaintext di log, seperti dan AWS CloudTrail Amazon CloudWatch Logs.

AWS Proton tidak mengambil konteks enkripsi yang ditentukan pelanggan atau yang ditentukan secara eksternal.

AWS Proton menambahkan konteks enkripsi berikut.

```
{
  "aws:proton:template": "<proton-template-arn>",
  "aws:proton:resource": "<proton-resource-arn>"
}
```

Konteks enkripsi pertama mengidentifikasi AWS Proton template yang terkait dengan sumber daya dan juga berfungsi sebagai kendala untuk izin dan hibah kunci yang dikelola pelanggan.

Konteks enkripsi kedua mengidentifikasi AWS Proton sumber daya yang dienkripsi.

Contoh berikut menunjukkan penggunaan konteks AWS Proton enkripsi.

Pengembang membuat instance layanan.

```
{
  "aws:proton:template": "arn:aws:proton:region_id:123456789012:service-template/my-
template",
  "aws:proton:resource": "arn:aws:proton:region_id:123456789012:service/my-service/
service-instance/my-service-instance"
}
```

Administrator membuat template.

```
{
```

```
  "aws:proton:template": "arn:aws:proton:region_id:123456789012:service-template/my-
template",
  "aws:proton:resource": "arn:aws:proton:region_id:123456789012:service-template/my-
template"
}
```

Keamanan infrastruktur di AWS Proton

Sebagai layanan terkelola, AWS Proton dilindungi oleh keamanan jaringan AWS global. Untuk informasi tentang layanan AWS keamanan dan cara AWS melindungi infrastruktur, lihat Keamanan AWS Cloud. Untuk mendesain AWS lingkungan Anda menggunakan praktik terbaik untuk keamanan infrastruktur, lihat Perlindungan Infrastruktur dalam Kerangka Kerja yang AWS Diarsiteksikan dengan Baik Pilar Keamanan.

Anda menggunakan panggilan API yang AWS dipublikasikan untuk mengakses AWS Proton melalui jaringan. Klien harus mendukung hal-hal berikut:

- Keamanan Lapisan Pengangkutan (TLS). Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Sandi cocok dengan sistem kerahasiaan maju sempurna (perfect forward secrecy, PFS) seperti DHE (Ephemeral Diffie-Hellman) atau ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Sebagian besar sistem modern seperti Java 7 dan versi lebih baru mendukung mode-mode ini.

Untuk meningkatkan isolasi jaringan, Anda dapat menggunakan AWS PrivateLink seperti yang dijelaskan di bagian berikut.

AWS Proton dan antarmuka titik akhir VPC ()AWS PrivateLink

Anda dapat membuat koneksi pribadi antara VPC Anda dan AWS Proton dengan membuat antarmuka VPC endpoint. Endpoint antarmuka didukung oleh AWS PrivateLink, teknologi yang memungkinkan Anda mengakses secara pribadi AWS Proton APIs tanpa gateway internet, perangkat NAT, koneksi VPN, atau koneksi. AWS Direct Connect Instans di VPC Anda tidak memerlukan alamat IP publik untuk berkomunikasi. AWS Proton APIs Lalu lintas antara VPC Anda dan AWS Proton tidak meninggalkan jaringan Amazon.

Setiap titik akhir antarmuka diwakili oleh satu atau beberapa Antarmuka Jaringan Elastis di subnet Anda.

Untuk informasi selengkapnya, lihat Antarmuka VPC endpoint (AWS PrivateLink) dalam Panduan Pengguna Amazon VPC.

Pertimbangan untuk titik akhir AWS Proton VPC

Sebelum menyiapkan titik akhir VPC antarmuka AWS Proton, pastikan Anda meninjau properti dan batasan titik akhir Antarmuka di Panduan Pengguna Amazon VPC.

AWS Proton mendukung panggilan ke semua tindakan API-nya dari VPC Anda.

Kebijakan titik akhir VPC didukung untuk. AWS Proton Secara default, akses penuh ke AWS Proton diizinkan melalui titik akhir. Untuk informasi selengkapnya, lihat Mengontrol Akses ke Layanan dengan titik akhir VPC dalam Panduan Pengguna Amazon VPC.

Membuat titik akhir VPC antarmuka untuk AWS Proton

Anda dapat membuat titik akhir VPC untuk AWS Proton layanan menggunakan konsol VPC Amazon atau (). AWS Command Line Interface AWS CLI Untuk informasi selengkapnya, lihat Membuat titik akhir antarmuka dalam Panduan Pengguna Amazon VPC.

Buat titik akhir VPC untuk AWS Proton menggunakan nama layanan berikut:

- com.amazonaws. *region*.proton

Jika Anda mengaktifkan DNS pribadi untuk titik akhir, Anda dapat membuat permintaan API untuk AWS Proton menggunakan nama DNS default untuk Wilayah, misalnya,. `proton.`*region*`.amazonaws.com`

Untuk informasi selengkapnya, lihat Mengakses layanan melalui titik akhir antarmuka dalam Panduan Pengguna Amazon VPC.

Membuat kebijakan titik akhir VPC untuk AWS Proton

Anda dapat melampirkan kebijakan titik akhir ke VPC endpoint yang mengendalikan akses ke AWS Proton. Kebijakan titik akhir menentukan informasi berikut:

- Prinsipal yang dapat melakukan tindakan.
- Tindakan yang dapat dilakukan.
- Sumber daya yang menjadi target tindakan.

Untuk informasi selengkapnya, lihat Mengontrol Akses ke Layanan dengan titik akhir VPC dalam Panduan Pengguna Amazon VPC.

Contoh: Kebijakan titik akhir VPC untuk tindakan AWS Proton

Berikut ini adalah contoh kebijakan endpoint untuk AWS Proton. Saat dilampirkan ke titik akhir, kebijakan ini memberikan akses ke AWS Proton tindakan yang tercantum untuk semua prinsipal di semua sumber daya.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Principal": "*",
      "Action": [
        "proton:ListServiceTemplates",
        "proton:ListServiceTemplateMajorVersions",
        "proton:ListServiceTemplateMinorVersions",
        "proton:ListServices",
        "proton:ListServiceInstances",
        "proton:ListEnvironments",
        "proton:GetServiceTemplate",
        "proton:GetServiceTemplateMajorVersion",
        "proton:GetServiceTemplateMinorVersion",
        "proton:GetService",
        "proton:GetServiceInstance",
        "proton:GetEnvironment",
        "proton:CreateService",
        "proton:UpdateService",
        "proton:UpdateServiceInstance",
        "proton:UpdateServicePipeline",
        "proton:DeleteService"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Penebangan dan pemantauan di AWS Proton

Pemantauan adalah bagian penting dari menjaga keandalan, ketersediaan, dan kinerja AWS Proton dan AWS solusi Anda yang lain. AWS menyediakan alat pemantauan berikut untuk melihat instans Anda berjalan AWS Proton, melaporkan ketika ada sesuatu yang salah, dan mengambil tindakan otomatis bila perlu.

Pada saat ini, AWS Proton dirinya sendiri tidak terintegrasi dengan Amazon CloudWatch Logs atau AWS Trusted Advisor. Administrator dapat mengkonfigurasi dan menggunakan CloudWatch untuk memantau lainnya Layanan AWS sebagaimana didefinisikan dalam layanan dan lingkungan template mereka. AWS Proton terintegrasi dengan AWS CloudTrail.

- Amazon CloudWatch memantau AWS sumber daya Anda dan aplikasi yang Anda jalankan AWS secara real time. Anda dapat mengumpulkan dan melacak metrik, membuat dasbor yang disesuaikan, dan mengatur alarm yang memberi tahu Anda atau mengambil tindakan saat metrik tertentu mencapai ambang batas yang ditentukan. Misalnya, Anda dapat CloudWatch melacak penggunaan CPU atau metrik lain dari EC2 instans Amazon Anda dan secara otomatis meluncurkan instans baru bila diperlukan. Untuk informasi selengkapnya, lihat Panduan CloudWatch Pengguna Amazon.
- Amazon CloudWatch Logs memungkinkan Anda memantau, menyimpan, dan mengakses file log Anda dari EC2 instans Amazon CloudTrail, dan sumber lainnya. CloudWatch Log dapat memantau informasi dalam file log dan memberi tahu Anda ketika ambang batas tertentu terpenuhi. Anda juga dapat mengarsipkan data log dalam penyimpanan yang sangat durabel. Untuk informasi selengkapnya, lihat Panduan Pengguna Amazon CloudWatch Logs.
- AWS CloudTrailmenangkap panggilan API dan peristiwa terkait yang dibuat oleh atau atas nama Anda Akun AWS dan mengirimkan file log ke bucket Amazon S3 yang Anda tentukan. Anda dapat mengidentifikasi pengguna dan akun mana yang dipanggil AWS, alamat IP sumber dari mana panggilan dilakukan, dan kapan panggilan terjadi. Untuk informasi selengkapnya, silakan lihat Panduan Pengguna AWS CloudTrail.
- Amazon EventBridge adalah layanan bus acara tanpa server yang memudahkan untuk menghubungkan aplikasi Anda dengan data dari berbagai sumber. EventBridge mengirimkan aliran data real-time dari aplikasi Anda sendiri, aplikasi Software-as-a-Service (SaaS), Layanan AWS dan dan merutekan data tersebut ke target seperti Lambda. Hal ini memungkinkan Anda memantau kejadian yang terjadi dalam layanan, dan membangun arsitektur yang didorong kejadian. Untuk informasi selengkapnya, lihat Otomatisasi dengan AWS Proton EventBridge dan Panduan Pengguna EventBridge .

Ketahanan di AWS Proton

Infrastruktur AWS global dibangun di sekitar AWS Region dan Availability Zones. AWS Region s menyediakan beberapa Availability Zone yang terpisah secara fisik dan terisolasi, yang terhubung dengan latensi rendah, throughput tinggi, dan jaringan yang sangat redundan. Dengan Zona Ketersediaan, Anda dapat merancang serta mengoperasikan aplikasi dan basis data yang secara otomatis melakukan fail over di antara zona tanpa gangguan. Zona Ketersediaan memiliki ketersediaan dan toleransi kesalahan yang lebih baik, dan dapat diskalakan dibandingkan infrastruktur pusat data tunggal atau multi tradisional.

Untuk informasi selengkapnya tentang AWS Region s dan Availability Zone, lihat Infrastruktur AWS Global.

Selain infrastruktur AWS global, AWS Proton menawarkan fitur untuk membantu mendukung ketahanan data dan kebutuhan cadangan Anda.

AWS Proton cadangan

AWS Proton memelihara cadangan semua data pelanggan. Dalam kasus pemadaman total, cadangan ini dapat digunakan untuk memulihkan AWS Proton dan data pelanggan dari keadaan valid sebelumnya.

Praktik terbaik keamanan untuk AWS Proton

AWS Proton menyediakan 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 tidak memadai untuk lingkungan Anda, perlakukan itu sebagai pertimbangan yang bermanfaat, bukan sebagai resep.

Topik

- Gunakan IAM untuk mengontrol akses
- Jangan menanamkan kredensi di template dan bundel template Anda
- Gunakan enkripsi untuk melindungi data sensitif
- Gunakan AWS CloudTrail untuk melihat dan mencatat panggilan API

Gunakan IAM untuk mengontrol akses

IAM adalah Layanan AWS yang dapat Anda gunakan untuk mengelola pengguna dan izin mereka. AWS Anda dapat menggunakan IAM AWS Proton untuk menentukan AWS Proton tindakan yang dapat dilakukan oleh administrator dan pengembang, seperti mengelola templat, lingkungan, atau layanan. Anda dapat menggunakan peran layanan IAM untuk memungkinkan AWS Proton melakukan panggilan ke layanan lain atas nama Anda.

Untuk informasi lebih lanjut tentang AWS Proton dan peran IAM, lihatIdentity and Access Management untuk AWS Proton.

Menerapkan akses hak istimewa paling sedikit. Untuk informasi selengkapnya, lihat Kebijakan dan izin di IAM di AWS Identity and Access Management Panduan Pengguna.

Jangan menanamkan kredensi di template dan bundel template Anda

Daripada menyematkan informasi sensitif dalam CloudFormation template dan bundel template Anda, kami sarankan Anda menggunakan referensi dinamis dalam template tumpukan Anda.

Referensi dinamis menyediakan cara yang ringkas dan ampuh bagi Anda untuk mereferensikan nilai eksternal yang disimpan dan dikelola di layanan lain, seperti AWS Systems Manager Parameter Store atau AWS Secrets Manager. Bila Anda menggunakan referensi dinamis, CloudFormation mengambil nilai referensi yang ditentukan bila diperlukan selama tumpukan dan mengubah operasi set, dan meneruskan nilai ke sumber daya yang sesuai. Namun, CloudFormation tidak pernah menyimpan nilai referensi yang sebenarnya. Untuk informasi selengkapnya, lihat Menggunakan Referensi Dinamis untuk Menentukan Nilai Template di Panduan CloudFormation Pengguna.

AWS Secrets Managermembantu Anda mengenkripsi, menyimpan, dan mengambil kredensil untuk database dan layanan lainnya dengan aman. AWS Systems Manager Parameter Store menyediakan penyimpanan hierarkis yang aman untuk manajemen data konfigurasi.

Untuk informasi selengkapnya tentang menentukan parameter template, lihat https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/parameters-section-structure.html di Panduan CloudFormation Pengguna.

Gunakan enkripsi untuk melindungi data sensitif

Di dalam AWS Proton, semua data pelanggan dienkripsi secara default menggunakan kunci yang AWS Proton dimiliki.

Sebagai anggota tim platform, Anda dapat memberikan kunci yang dikelola pelanggan AWS Proton untuk mengenkripsi dan mengamankan data sensitif Anda. Enkripsi data sensitif saat istirahat di bucket S3 Anda. Untuk informasi selengkapnya, lihat Perlindungan data di AWS Proton.

Gunakan AWS CloudTrail untuk melihat dan mencatat panggilan API

AWS CloudTrail melacak siapa pun yang melakukan panggilan API di Anda Akun AWS. Panggilan API dicatat setiap kali ada yang menggunakan AWS Proton API, AWS Proton konsol, atau AWS Proton AWS CLI perintah. Aktifkan logging dan tentukan bucket Amazon S3 untuk menyimpan log. Dengan begitu, jika perlu, Anda dapat mengaudit siapa yang melakukan AWS Proton panggilan apa di akun Anda. Lihat informasi yang lebih lengkap di Penebangan dan pemantauan di AWS Proton.

Pencegahan "confused deputy" lintas layanan

Masalah "confused deputy" adalah masalah keamanan di mana entitas yang tidak memiliki izin untuk melakukan tindakan dapat memengaruhi entitas yang memiliki hak akses lebih tinggi untuk melakukan tindakan. Pada tahun AWS, peniruan lintas layanan dapat mengakibatkan masalah wakil yang membingungkan. Peniruan identitas lintas layanan dapat terjadi ketika satu layanan (layanan yang dipanggil) memanggil layanan lain (layanan yang dipanggil). Layanan pemanggilan dapat dimanipulasi menggunakan izinnya untuk bertindak pada sumber daya pelanggan lain dengan cara yang seharusnya tidak dilakukannya kecuali bila memiliki izin untuk mengakses. Untuk mencegah hal ini, AWS sediakan alat yang membantu Anda melindungi data Anda untuk semua layanan dengan prinsip layanan yang telah diberikan akses ke sumber daya di akun Anda.

Sebaiknya gunakan kunci konteks kondisi `aws:SourceAccount`global `aws:SourceArn`dan dalam kebijakan sumber daya untuk membatasi izin yang AWS Proton memberikan layanan lain ke sumber daya. Jika nilai `aws:SourceArn` tidak berisi ID akun, seperti ARN bucket Amazon S3, Anda harus menggunakan kedua kunci konteks kondisi global tersebut untuk membatasi izin. Jika Anda menggunakan kunci konteks kondisi global dan nilai `aws:SourceArn` berisi ID akun, nilai `aws:SourceAccount` dan akun dalam nilai `aws:SourceArn` harus menggunakan ID akun yang sama saat digunakan dalam pernyataan kebijakan yang sama. Gunakan `aws:SourceArn` jika Anda ingin hanya satu sumber daya yang akan dikaitkan dengan akses lintas layanan. Gunakan `aws:SourceAccount` jika Anda ingin mengizinkan sumber daya apa pun di akun tersebut dikaitkan dengan penggunaan lintas layanan.

Nilai `aws:SourceArn` harus menjadi sumber daya yang AWS Proton menyimpan.

Cara paling efektif untuk melindungi dari masalah "confused deputy" adalah dengan menggunakan kunci konteks kondisi global `aws:SourceArn` dengan ARN lengkap sumber daya. Jika Anda tidak

mengetahui ARN lengkap sumber daya atau jika Anda menentukan beberapa sumber daya, gunakan kunci kondisi konteks `aws:SourceArn` global dengan wildcard (*) untuk bagian ARN yang tidak diketahui. Misalnya, `arn:aws::proton:*:123456789012:environment/*`.

Contoh berikut menunjukkan bagaimana Anda dapat menggunakan kunci konteks kondisi `aws:SourceAccount` global `aws:SourceArn` dan AWS Proton untuk mencegah masalah wakil yang membingungkan.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Sid": "ExampleProtonConfusedDeputyPreventionPolicy",
        "Effect": "Allow",
        "Principal": {"Service": "proton.amazonaws.com"},
        "Action": "sts:AssumeRole",
        "Condition": {
            "StringEquals": {
                "aws:SourceAccount": "123456789012"
            },
            "ArnLike": {
                "aws:SourceArn": "arn:aws:proton:*:123456789012:environment/*"
            }
        }
    }
}
```

CodeBuild penyediaan dukungan VPC Amazon khusus

AWS Proton CodeBuild Penyediaan mengeksekusi perintah CLI yang disediakan pelanggan sewenang-wenang dalam proyek yang terletak di CodeBuild akun Lingkungan. AWS Proton Perintah ini biasanya mengelola sumber daya menggunakan alat Infrastructure as Code (IAc), seperti CDK. Jika Anda memiliki sumber daya di VPC Amazon, CodeBuild mungkin tidak dapat mengaksesnya. Untuk mengaktifkan ini, CodeBuild mendukung kemampuan untuk berjalan dalam VPC Amazon tertentu. Beberapa contoh kasus penggunaan meliputi:

- Ambil dependensi dari repositori artefak internal yang dihosting sendiri, seperti `PyPI` untuk Python, untuk Java, dan untuk Node.js `Maven` `npm`

- CodeBuild perlu mengakses server Jenkins di VPC Amazon tertentu untuk mendaftarkan pipeline.
- Akses objek dalam bucket Amazon S3 yang dikonfigurasi untuk mengizinkan akses melalui titik akhir VPC Amazon saja.
- Jalankan pengujian integrasi dari build Anda terhadap data dalam database Amazon RDS yang terisolasi di subnet pribadi.

Untuk informasi selengkapnya, lihat CodeBuild dan dokumentasi VPC.

Jika Anda ingin CodeBuild Provisioning berjalan di VPC kustom, AWS Proton berikan solusi langsung. Pertama, Anda harus menambahkan ID VPC, subnet, dan grup keamanan ke template lingkungan. Selanjutnya, Anda memasukkan nilai-nilai tersebut ke dalam spesifikasi lingkungan. Ini akan menghasilkan CodeBuild proyek yang dibuat untuk Anda yang menargetkan VPC tertentu.

Memperbarui Template Lingkungan

Skema

ID VPC, subnet, dan grup keamanan perlu ditambahkan ke skema template sehingga mereka dapat ada dalam spesifikasi lingkungan.

Contoh`schema.yaml`:

```
schema:
  format:
    openapi: "3.0.0"
  environment_input_type: "EnvironmentInputType"
  types:
    EnvironmentInputType:
      type: object
      properties:
        codebuild_vpc_id:
          type: string
        codebuild_subnets:
          type: array
          items:
            type: string
        codebuild_security_groups:
          type: array
          items:
            type: string
```

Ini menambahkan tiga properti baru yang akan digunakan oleh manifes:

- `codebuild_vpc_id`
- `codebuild_subnets`
- `codebuild_security_groups`

Manifes

Untuk mengonfigurasi setelan VPC Amazon CodeBuild, properti opsional yang disebut `project_properties` tersedia di manifes templat. Isi `project_properties` ditambahkan ke CloudFormation tumpukan yang membuat CodeBuild proyek. Ini memungkinkan untuk menambahkan tidak hanya properti VPC Amazon, tetapi juga CloudFormation properti apa pun yang didukung CodeBuild CloudFormation , seperti batas waktu build. Data yang sama `proton-inputs.json` disediakan untuk dibuat tersedia untuk nilai-nilai`project_properties`.

Tambahkan bagian ini ke`manifest.yaml`:

```
project_properties:
          VpcConfig:
            VpcId: "{{ environment.inputs.codebuild_vpc_id }}"
            Subnets: "{{ environment.inputs.codebuild_subnets }}"
            SecurityGroupIds: "{{ environment.inputs.codebuild_security_groups }}"
```

Berikut ini adalah seperti apa hasilnya`manifest.yaml`:

```
infrastructure:
  templates:
    - rendering_engine: codebuild
      settings:
        image: aws/codebuild/amazonlinux2-x86_64-standard:4.0
        runtimes:
          nodejs: 16
        provision:
          - npm install
          - npm run build
          - npm run cdk bootstrap
          - npm run cdk deploy -- --require-approval never
        deprovision:
          - npm install
          - npm run build
```

```
          - npm run cdk destroy -- --force
        project_properties:
          VpcConfig:
            VpcId: "{{ environment.inputs.codebuild_vpc_id }}"
            Subnets: "{{ environment.inputs.codebuild_subnets }}"
            SecurityGroupIds: "{{ environment.inputs.codebuild_security_groups }}"
```

Menciptakan lingkungan

Saat membuat lingkungan dengan template yang mendukung VPC CodeBuild Penyediaan, Anda harus memberikan ID VPC Amazon, subnet, dan grup keamanan.

Untuk mendapatkan daftar semua VPC Amazon IDs di Wilayah Anda, jalankan perintah berikut:

```
aws ec2 describe-vpcs
```

Untuk mendapatkan daftar semua subnet IDs, jalankan:

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=vpc-id"
```

> **Important**
>
> Hanya sertakan subnet pribadi. CodeBuild akan gagal jika Anda menyediakan subnet publik. Subnet publik memiliki rute default ke Internet Gateway, sedangkan subnet pribadi tidak.

Jalankan perintah berikut untuk mendapatkan grup keamanan IDs. Ini juga IDs dapat diperoleh melalui Konsol Manajemen AWS:

```
aws ec2 describe-security-groups --filters "Name=vpc-id,Values=vpc-id"
```

Nilainya akan menyerupai:

```
vpc-id: vpc-045ch35y28dec3a05
subnets:
  - subnet-04029a82e6ae46968
  - subnet-0f500a9294fc5f26a
 security-groups:
  - sg-03bc4c4ce32d67e8d
```

Memastikan CodeBuild izin

Dukungan Amazon VPC memerlukan izin tertentu, seperti kemampuan untuk membuat Antarmuka Jaringan Elastis.

Jika lingkungan sedang dibuat di konsol, masukkan nilai-nilai ini selama wizard pembuatan lingkungan. Jika Anda ingin membuat lingkungan secara terprogram, `spec.yaml` penampilan Anda seperti berikut:

```
proton: EnvironmentSpec

spec:
  codebuild_vpc_id: vpc-045ch35y28dec3a05
  codebuild_subnets:
    - subnet-04029a82e6ae46968
    - subnet-0f500a9294fc5f26a
  codebuild_security_groups:
    - sg-03bc4c4ce32d67e8d
```

AWS Proton sumber daya dan penandaan

AWS Proton sumber daya yang diberi Nama Sumber Daya Amazon (ARN) mencakup templat lingkungan dan versi mayor dan minor, templat layanan dan versi mayor dan minor, lingkungan, layanan, instance layanan, komponen, dan repositori. Anda dapat menandai sumber daya ini untuk membantu Anda mengatur dan mengidentifikasi mereka. Anda dapat menggunakan tag untuk mengategorikan sumber daya berdasarkan tujuan, pemilik, lingkungan, atau kriteria lainnya. Untuk informasi selengkapnya, lihat Strategi Penandaan . Untuk melacak dan mengelola AWS Proton sumber daya, Anda dapat menggunakan fitur penandaan yang dijelaskan di bagian berikut.

AWS penandaan

Anda dapat menetapkan metadata ke AWS sumber daya Anda dalam bentuk tag. Setiap tag terdiri dari kunci yang ditentukan pelanggan dan nilai opsional. Tanda membantu Anda mengelola, mengidentifikasi, mengatur, dan memfilter sumber daya.

> **Important**
>
> Jangan menambahkan informasi pengenal pribadi (PII) atau informasi rahasia atau sensitif lainnya dalam tag. Tag dapat diakses oleh banyak orang Layanan AWS, termasuk penagihan. Tag tidak dimaksudkan untuk digunakan dalam data sensitif atau privat.

Setiap tag memiliki dua bagian.

- Kunci tag (misalnya,, `CostCenterEnvironment`, atau`Project`). Kunci tag peka huruf besar dan kecil.
- Nilai tag (opsional) (misalnya, `111122223333` atau`Production`). Seperti kunci tag, nilai tag peka huruf besar dan kecil.

Persyaratan penamaan dan penggunaan dasar berikut berlaku untuk tag.

- Setiap sumber daya dapat memiliki maksimum 50 tag yang dibuat pengguna.

> **Note**
>
> Tag yang dibuat sistem yang dimulai dengan `aws:` awalan dicadangkan untuk AWS digunakan, dan tidak dihitung terhadap batas ini. Anda tidak dapat mengedit atau menghapus tag yang dimulai dengan prefiks `aws:`.

- Untuk setiap sumber daya, setiap kunci tag harus unik, dan setiap kunci tag hanya dapat memiliki satu nilai.
- Kunci tag harus minimal 1 dan maksimal 128 karakter Unicode dalam UTF-8.
- Nilai tag harus minimal 1 dan maksimal 256 karakter Unicode di UTF-8.
- Karakter yang diizinkan dalam tag adalah huruf, angka, spasi yang dapat direpresentasikan dalam UTF-8, dan karakter berikut: * _.:/= + - @.

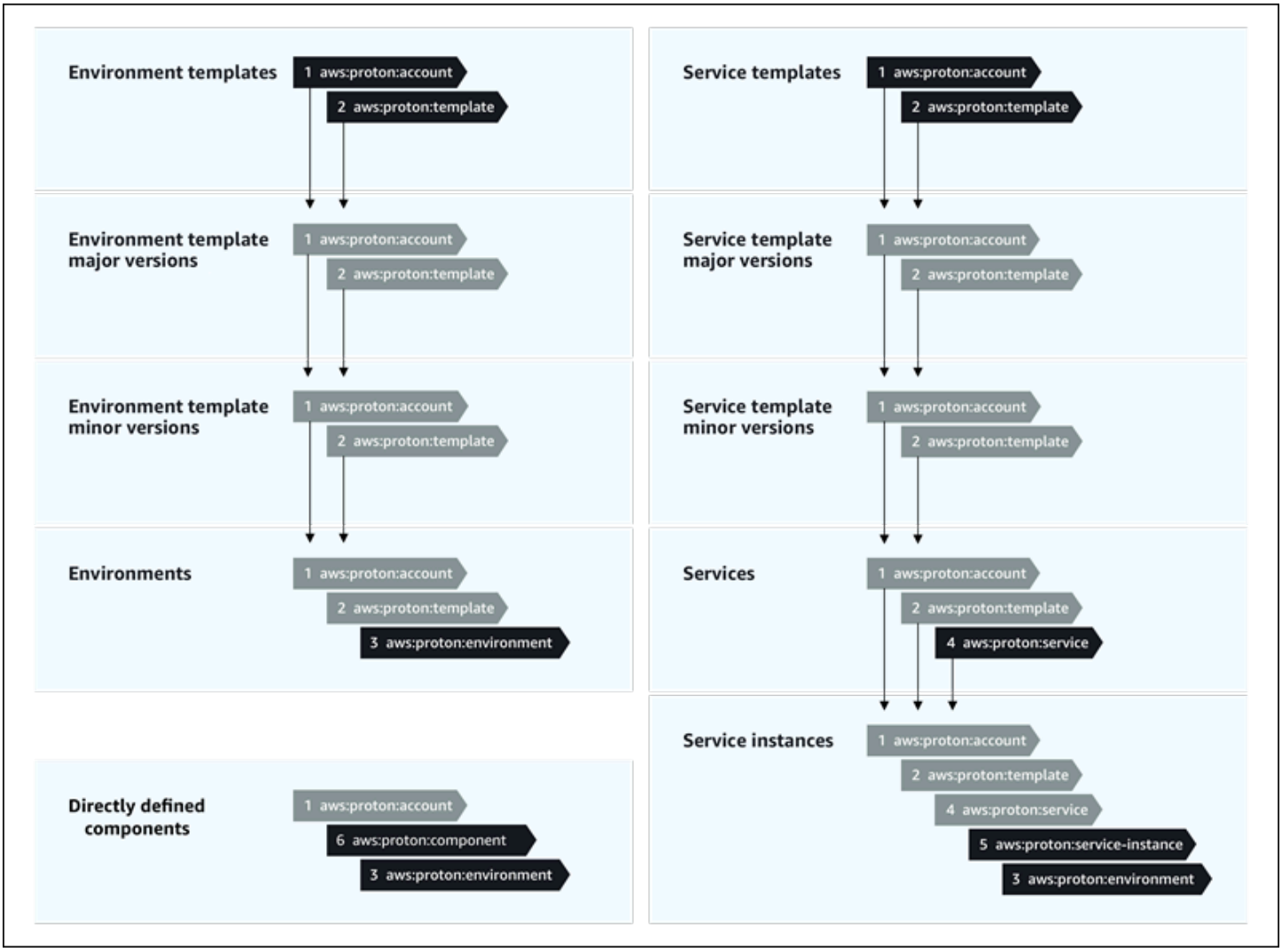
AWS Proton penandaan

Dengan AWS Proton, Anda dapat menggunakan kedua tag yang Anda buat serta tag yang AWS Proton secara otomatis menghasilkan untuk Anda.

AWS Proton AWS tag terkelola

Saat Anda membuat AWS Proton sumber daya, AWS Proton secara otomatis menghasilkan tag AWS terkelola untuk sumber daya baru Anda seperti yang ditunjukkan pada diagram berikut. AWS tag terkelola kemudian menyebar ke AWS Proton sumber daya lain yang didasarkan pada sumber daya baru Anda. Misalnya, tag terkelola dari templat lingkungan menyebar ke versinya, dan tag terkelola dari layanan menyebar ke instance layanannya.

> **Note**
>
> AWS tag terkelola tidak dibuat untuk koneksi akun lingkungan. Untuk informasi selengkapnya, lihat the section called “Koneksi akun”.



Perbanyakan tag ke sumber daya yang disediakan

Jika sumber daya yang disediakan, seperti yang didefinisikan dalam templat layanan dan lingkungan, mendukung penandaan, AWS tag terkelola akan menyebar sebagai tag yang AWS dikelola pelanggan ke sumber daya yang disediakan. Tag ini tidak akan menyebar ke sumber daya yang disediakan yang tidak mendukung penandaan. AWS

AWS Proton menerapkan tag ke sumber daya Anda berdasarkan AWS Proton akun, templat terdaftar, dan lingkungan yang diterapkan, serta layanan dan instance layanan seperti yang dijelaskan dalam tabel berikut. Anda dapat menggunakan tag AWS terkelola untuk melihat dan mengelola AWS Proton sumber daya Anda, tetapi Anda tidak dapat memodifikasinya.

AWS kunci tag terkelola	Kunci yang dikelola pelanggan yang disebarkan	Deskripsi
`aws:proton:account`	`proton:account`	AWS Akun yang membuat dan menyebarkan AWS Proton sumber daya.
`aws:proton:template`	`proton:template`	ARN dari template yang dipilih.
`aws:proton:environment`	`proton:environment`	ARN dari lingkungan yang dipilih.
`aws:proton:service`	`proton:service`	ARN dari layanan yang dipilih.
`aws:proton:service-instance`	`proton:service-instance`	ARN dari instance layanan yang dipilih.
`aws:proton:component`	`proton:component`	ARN dari komponen yang dipilih.

Berikut ini adalah contoh tag AWS terkelola untuk AWS Proton sumber daya.

```
"aws:proton:template" = "arn:aws:proton:region-id:account-id:environment-template/env-template"
```

Berikut ini adalah contoh tag terkelola pelanggan yang diterapkan ke sumber daya yang disediakan yang disebarkan dari tag terkelola AWS .

```
"proton:environment:database" = "arn:aws:proton:region-id:account-id:rds/env-db"
```

Dengan penyediaan AWS-managed, AWS Proton menerapkan tag yang disebarkan langsung ke sumber daya yang disediakan.

Dengan penyediaan yang dikelola sendiri, AWS Proton membuat tag yang disebarkan tersedia bersama dengan file IAc yang dirender yang dikirimkan dalam permintaan tarik penyediaan (PR). Tag disediakan dalam variabel peta string`proton_tags`. Kami menyarankan Anda membuat

referensi ke variabel ini dalam konfigurasi Terraform Anda untuk menyertakan AWS Proton tag di. `default_tags` Ini menyebarkan AWS Proton tag ke semua sumber daya yang disediakan.

Contoh berikut menunjukkan metode propagasi tag ini di lingkungan Template Terraform.

Berikut definisi `proton_tags` variabelnya:

proton.environment.variables.tf:

```
variable "environment" {
  type = object({
    inputs = map(string)
    name = string
  })
}

variable "proton_tags" {
  type = map(string)
  default = null
}
```

Berikut cara nilai tag ditetapkan ke variabel ini:

proton.auto.tfvars.json:

```
{
  "environment": {
    "name": "dev",
    "inputs": {
      "ssm_parameter_value": "MyNewParamValue"
    }
  }

  "proton_tags" : {
    "proton:account" : "123456789012",
    "proton:template" : "arn:aws:proton:us-east-1:123456789012:environment-template/
fargate-env",
    "proton:environment" : "arn:aws:proton:us-east-1:123456789012:environment/dev"
  }
}
```

Dan inilah cara Anda dapat menambahkan AWS Proton tag ke konfigurasi Terraform Anda sehingga ditambahkan ke sumber daya yang disediakan:

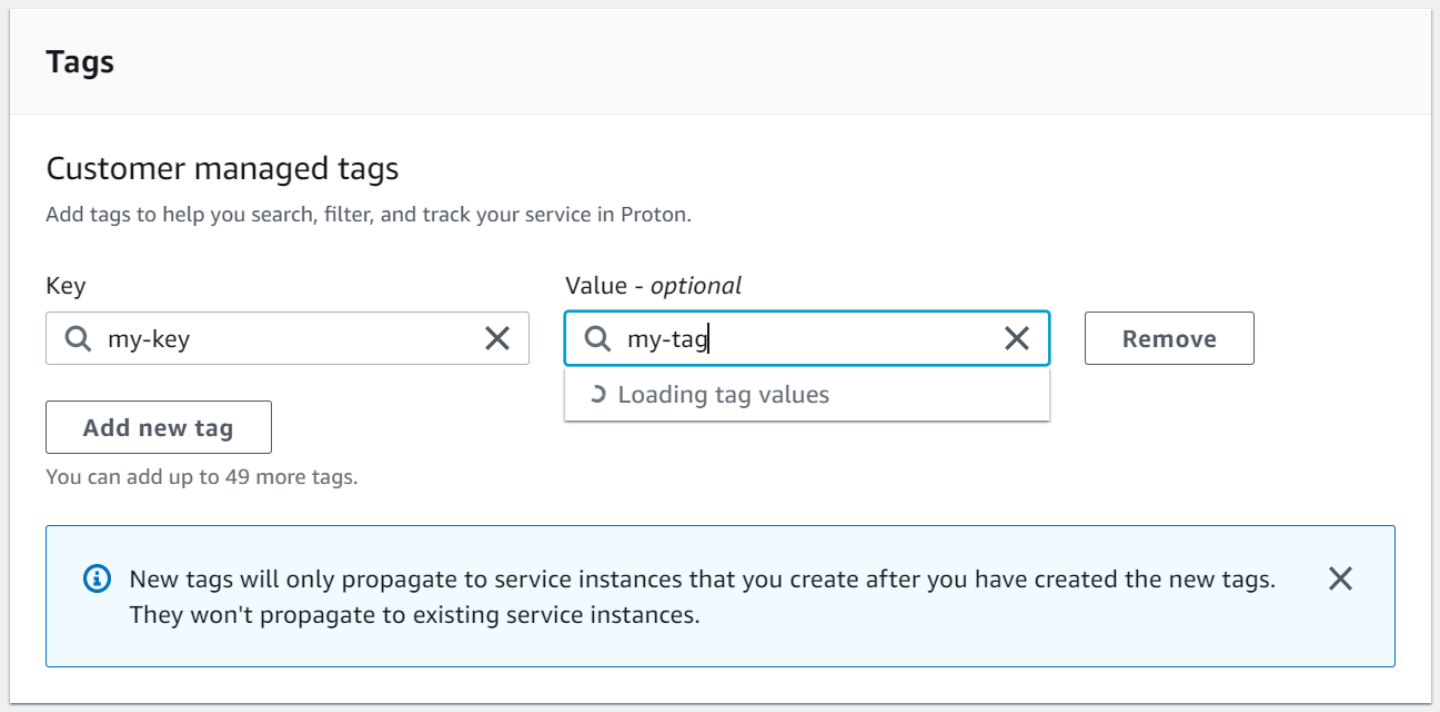
```
# Configure the AWS Provider
provider "aws" {
  region = var.aws_region
  default_tags {
    tags = var.proton_tags
  }
}
```

Tag terkelola pelanggan

Setiap AWS Proton sumber daya memiliki kuota maksimum 50 tag yang dikelola pelanggan. Tag terkelola pelanggan menyebar ke AWS Proton sumber daya anak dengan cara yang sama seperti tag AWS terkelola, kecuali tag tersebut tidak menyebar ke AWS Proton sumber daya yang ada atau ke sumber daya yang disediakan. Jika Anda menerapkan tag baru ke AWS Proton sumber daya dengan sumber daya turunan yang ada dan Anda ingin sumber daya anak yang ada diberi tag dengan tag baru, Anda perlu menandai setiap sumber daya anak yang ada secara manual, menggunakan konsol atau AWS CLI.

Buat tag menggunakan konsol dan CLI

Saat membuat AWS Proton resource menggunakan konsol, Anda diberi kesempatan untuk membuat tag terkelola pelanggan baik di halaman pertama atau kedua prosedur pembuatan seperti yang ditunjukkan pada snapshot konsol berikut. Pilih Tambahkan tag baru, masukkan kunci dan nilai dan lanjutkan.



Setelah Anda membuat sumber daya baru menggunakan AWS Proton konsol, Anda dapat melihat daftar tag yang AWS dikelola dan dikelola pelanggan dari halaman detail.

Membuat atau mengedit tag

1. Di AWS Proton konsol, buka halaman detail AWS Proton sumber daya tempat Anda dapat melihat daftar tag.
2. Pilih Kelola tanda.
3. Di halaman Kelola tag, Anda dapat melihat, membuat, menghapus, dan mengedit tag. Anda tidak dapat mengubah tag AWS terkelola yang tercantum di bagian atas. Namun, Anda dapat menambahkan dan memodifikasi tag yang dikelola pelanggan dengan bidang pengeditan, yang tercantum setelah tag AWS terkelola.

   Pilih Tambahkan tag baru untuk membuat tag baru.
4. Masukkan kunci dan nilai untuk tag baru.
5. Untuk mengedit tag, masukkan nilai di bidang nilai tag untuk kunci yang dipilih.
6. Untuk menghapus tag, pilih Hapus untuk tag yang dipilih.
7. Setelah Anda menyelesaikan perubahan, pilih Simpan perubahan.

Buat tag menggunakan AWS Proton AWS CLI

Anda dapat melihat, membuat, menghapus, dan mengedit tag menggunakan AWS Proton AWS CLI.

Anda dapat membuat atau mengedit tag untuk sumber daya seperti yang ditunjukkan pada contoh berikut.

```
$ aws proton tag-resource \
    --resource-arn "arn:aws:proton:region-id:account-id:service-template/webservice" \
    --tags '[{"key":"mykey1","value":"myval1"},{"key":"mykey2","value":"myval2"}]'
```

Anda dapat menghapus tag untuk sumber daya seperti yang ditunjukkan pada contoh berikutnya.

```
$ aws proton untag-resource \
    --resource-arn "arn:aws:proton:region-id:account-id:service-template/webservice" \
    --tag-keys '["mykey1","mykey2"]'
```

Anda dapat mencantumkan tag untuk sumber daya seperti yang ditunjukkan pada contoh akhir.

```
$ aws proton list-tags-for-resource \
    --resource-arn "arn:aws:proton:region-id:account-id:service-template/webservice"
```

Pemecahan masalah AWS Proton

Belajarlah untuk memecahkan masalah dengan. AWS Proton

Topik

- Kesalahan penerapan yang mereferensikan parameter CloudFormation dinamis

Kesalahan penerapan yang mereferensikan parameter CloudFormation dinamis

Jika Anda melihat kesalahan penerapan yang mereferensikan variabel CloudFormation dinamis Anda, verifikasi bahwa mereka adalah Jinja yang lolos. Kesalahan ini dapat disebabkan oleh salah tafsir Jinja terhadap variabel dinamis Anda. Sintaks parameter CloudFormation dinamis sangat mirip dengan sintaks Jinja yang Anda gunakan dengan parameter Anda. AWS Proton

Contoh sintaks variabel CloudFormation dinamis:

`'{{resolve:secretsmanager:MySecret:SecretString:password:EXAMPLE1-90ab-cdef-fedc-ba987EXAMPLE}}'`.

Contoh AWS Proton parameter sintaks Jinja:

`'{{ service_instance.environment.outputs.env-outputs }}'`.

Untuk menghindari kesalahan salah tafsir ini, Jinja lolos dari parameter CloudFormation dinamis Anda seperti yang ditunjukkan pada contoh berikut.

Contoh ini dari Panduan CloudFormation Pengguna. Segmen AWS Secrets Manager secret-name dan json-key dapat digunakan untuk mengambil kredenal masuk yang disimpan dalam rahasia.

```
MyRDSInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    DBName: 'MyRDSInstance'
      AllocatedStorage: '20'
      DBInstanceClass: db.t2.micro
      Engine: mysql
      MasterUsername: '{{resolve:secretsmanager:MyRDSSecret:SecretString:username}}'
```

```
      MasterUserPassword:
 '{{resolve:secretsmanager:MyRDSSecret:SecretString:password}}'
```

Untuk menghindari parameter CloudFormation dinamis, Anda dapat menggunakan dua metode berbeda:

- Lampirkan blok antara`{% raw %}` and `{% endraw %}`:

```
'{% raw %}'
MyRDSInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    DBName: 'MyRDSInstance'
      AllocatedStorage: '20'
      DBInstanceClass: db.t2.micro
      Engine: mysql
      MasterUsername: '{{resolve:secretsmanager:MyRDSSecret:SecretString:username}}'
      MasterUserPassword:
 '{{resolve:secretsmanager:MyRDSSecret:SecretString:password}}'
'{% endraw %}'
```

- Lampirkan parameter antara`"{{ }}"`:

```
MyRDSInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    DBName: 'MyRDSInstance'
      AllocatedStorage: '20'
      DBInstanceClass: db.t2.micro
      Engine: mysql
      MasterUsername:
 "{{ '{{resolve:secretsmanager:MyRDSSecret:SecretString:username}}' }}"
      MasterUserPassword:
 "{{ '{{resolve:secretsmanager:MyRDSSecret:SecretString:password}}' }}"
```

Untuk informasi, lihat Jinja melarikan diri.

AWS Proton kuota

Tabel berikut mencantumkan AWS Proton kuota. Semua nilai adalah per AWS akun, per AWS Wilayah yang didukung.

Kuota sumber daya	Batas default	Dapat disesuaikan?
Ukuran maksimum bundel template	10 MB	✗ Tidak
Ukuran maksimum file manifes template	2 MB	✗ Tidak
Ukuran maksimum file skema template	2 MB	✗ Tidak
Ukuran maksimum setiap file template	2 MB	✗ Tidak
Panjang maksimum setiap nama template	100 karakter	✗ Tidak
Jumlah maksimum file CloudFormation template per bundel	1	✗ Tidak
Jumlah maksimum templat terdaftar per akun, layanan, dan templat lingkungan digabungkan	1000	✓ Ya
Jumlah maksimum versi template yang terdaftar per template	1000	✓ Ya
Jumlah maksimum file per bundel CodeBuild Penyediaan	500	✗ Tidak
Jumlah maksimum lingkungan per akun	1000	✓ Ya
Jumlah maksimum layanan per akun	1000	✓ Ya
Jumlah maksimum instans layanan per layanan	20	✓ Ya
Jumlah maksimum komponen per akun	1000	✓ Ya
Jumlah maksimum koneksi akun lingkungan per akun lingkungan	1000	✓ Ya

Riwayat dokumen

Tabel berikut menjelaskan perubahan penting pada dokumentasi yang terkait dengan rilis terbaru AWS Proton dan umpan balik pelanggan. Untuk notifikasi tentang pembaruan dokumentasi ini, Anda dapat berlangganan ke umpan RSS.

- Versi API: 2020-07-20

Perubahan	Deskripsi	Tanggal
Pemberitahuan akhir dukungan	Pada 7 Oktober 2026, AWS akan mengakhiri dukungan untuk AWS Proton.	Oktober 7, 2025
Pembaruan kebijakan terkelola	`AWSProtonCodeBuildProvisioningServiceRolePolicy` kebijakan telah diperbarui.	Juni 15, 2024
Pembaruan kebijakan terkelola	`AWSProtonDeveloperAccess` kebijakan telah diperbarui.	April 25, 2024
Pembaruan kebijakan terkelola	`AWSProtonFullAccess` kebijakan telah diperbarui.	April 25, 2024
Pembaruan kebijakan terkelola	`AWSProtonSyncServiceRolePolicy` kebijakan telah diperbarui.	April 25, 2024
Pembaruan kebijakan terkelola	`AWSProtonCodeBuildProvisioningServiceRolePolicy` kebijakan telah diperbarui.	12 Mei 2023

Konfigurasi sinkronisasi layanan.	AWS Proton menambahkan dukungan untuk konfigurasi sinkronisasi layanan.	31 Maret 2023
CodeBuild	AWS Proton menambahkan dukungan untuk CodeBuild penyediaan.	16 November 2022
Pembaruan kebijakan terkelola	Menambahkan `AWSProtonCodeBuildProvisioningBasicAccess` kebijakan CodeBuild yang memberikan izin yang diperlukan untuk menjalankan build untuk AWS Proton CodeBuild Penyediaan.	11 November 2022
Perbanyakan tag Terraform	Menambahkan propagasi tag Terraform ke chapter Tagging.	September 16, 2022
Panduan migrasi API	Menghapus panduan migrasi API pra-GA.	12 Agustus 2022
AWS Proton benda-benda	Menambahkan topik tentang AWS Proton objek dan hubungannya dengan objek lain AWS dan pihak ketiga. Lihat AWS Proton objek.	Juli 29, 2022
Klarifikasi repositori tertaut	Mengklarifikasi tujuan repositori tertaut (terdaftar) dan penggunaannya di seluruh panduan.	18 Juli 2022

Gabungan panduan	Menggabungkan dua administrator terpisah dan panduan pengguna ke dalam satu panduan, Panduan AWS Proton Pengguna.	30 Juni 2022
Pembaruan kebijakan terkelola	Kebijakan terkelola yang diperbarui untuk menyediakan akses AWS Proton ke operasi API baru dan untuk memperbaiki masalah izin untuk beberapa operasi AWS Proton konsol. Lihat kebijakan AWS terkelola untuk AWS Proton.	Juni 20, 2022
Memulai dengan CLI	Diperbarui Memulai AWS CLI dengan tutorial baru yang menggunakan pustaka template baru.	14 Juni 2022
Komponen yang didefinisikan secara langsung	Menambahkan bagian Komponen dan membuat modifikasi terkait di seluruh panduan.	1 Juni 2022
AWS Proton pustaka template	Ditambahkan Topik pustaka AWS Proton template.	6 Mei 2022
Ketersediaan umum Terraform (GA)	Mengganti nama penyediaan permintaan tarik menjadi penyediaan yang dikelola sendiri. Ditambahkan topik metode Provisioning.	Maret 23, 2022

Penandaan repositori	Menambahkan dukungan untuk menandai sumber daya Repositori. Lihat Membuat tautan ke repositori Anda.	Maret 23, 2022
Pembaruan dokumentasi	Menambahkan penandaan koneksi akun lingkungan.	26 November 2021
Sinkronisasi templat dan pratinjau Terraform	Menambahkan versi template otomatis dengan fitur sinkronisasi templat untuk ketersediaan umum dan penyediaan permintaan tarik dengan Terraform di pratinjau. Panduan migrasi API kembali masuk.	24 November 2021
Pembaruan dokumentasi	Menambahkan EventBridgetutorial, Memulai alur kerja, Cara AWS Proton kerja, dan penyempurnaan bagian bundel Template.	September 17, 2021
AWS Proton panel bantuan konsol rilis	Panel bantuan ditambahkan ke konsol. Versi template konsol menghapus tidak lagi menghapus versi yang lebih rendah. Panduan migrasi API telah dihapus.	8 September 2021
AWS Proton rilis ketersediaan umum (GA)	Menambahkan lingkungan lintas akun, EventBridge pemantauan, kunci kondisi IAM, dukungan idempotensi, dan peningkatan kuota.	9 Juni 2021

Menambahkan dan menghapus instance layanan untuk layanan dan menggunakan infrastruktur eksternal yang ada untuk lingkungan dengan AWS Proton	Rilis pratinjau publik ini mencakup pembaruan yang memungkinkan Anda menambahkan dan menghapus instance layanan dari layanan, menggunakan infrastruktur eksternal yang ada di lingkungan, dan untuk membatalkan AWS Proton lingkungan, instance layanan, dan penerapan pipeline. AWS Proton sekarang mendukung PrivateLink. Validasi penghapusan tambahan telah ditambahkan untuk mencegah versi minor dihapus secara keliru saat sumber daya menggunakannya.	27 April 2021
Menandai dengan AWS Proton	Rilis pratinjau publik 2 mencakup AWS Proton penandaan dan kemampuan untuk meluncurkan layanan tanpa saluran layanan.	5 Maret 2021
Rilis awal	Rilis pratinjau publik sekarang tersedia di wilayah tertentu.	1 Desember 2020

AWS Glosarium

Untuk AWS terminologi terbaru, lihat AWS glosarium di Referensi.Glosarium AWS