aws

User Guide

AWS Proton

AWS Proton: User Guide

Table of Contents

Hinweis zum Ende des Supports: Am 7. Oktober 2026 AWS endet der Support für AWS Proton. Nach dem 7. Oktober 2026 können Sie nicht mehr auf die AWS Proton Konsole oder AWS Proton die Ressourcen zugreifen. Ihre bereitgestellte Infrastruktur bleibt intakt. Weitere Informationen finden Sie im AWS Proton Service Deprecation and Migration Guide.

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.

Was ist AWS Proton?

AWS Proton ist:

- Automatisierte Infrastruktur als Codebereitstellung und Bereitstellung serverloser und containerbasierter Anwendungen

  Der AWS Proton Service ist ein zweigleisiges Automatisierungs-Framework. Als Administrator erstellen Sie versionierte Dienstvorlagen, die standardisierte Infrastruktur- und Bereitstellungstools für serverlose und containerbasierte Anwendungen definieren. Als Anwendungsentwickler können Sie aus den verfügbaren Dienstvorlagen auswählen, um Ihre Anwendungs- oder Servicebereitstellungen zu automatisieren.

  AWS Proton identifiziert für Sie alle vorhandenen Serviceinstanzen, die eine veraltete Vorlagenversion verwenden. Als Administrator können Sie mit einem Klick beantragen AWS Proton , dass sie aktualisiert werden.
- Standardisierte Infrastruktur

  Plattformteams können eine AWS Proton versionierte Infrastruktur als Codevorlagen verwenden. Sie können diese Vorlagen verwenden, um Standardanwendungsstapel zu definieren und zu verwalten, die die Architektur, die Infrastrukturressourcen und die Pipeline für die CI/CD Softwarebereitstellung enthalten.
- In CI/CD integrierte Bereitstellungen

  Wenn Entwickler die AWS Proton Self-Service-Schnittstelle verwenden, um eine Dienstvorlage auszuwählen, wählen sie eine standardisierte Anwendungsstapeldefinition für ihre Codebereitstellungen aus. AWS Proton stellt automatisch die Ressourcen bereit, konfiguriert die CI/CD Pipeline und stellt den Code in der definierten Infrastruktur bereit.

AWS Proton für Plattformteams

Als Administrator können Sie oder Mitglieder Ihres Plattformteams Umgebungs- und Dienstvorlagen erstellen, die Infrastruktur als Code enthalten. Die Umgebungsvorlage definiert eine gemeinsam genutzte Infrastruktur, die von mehreren Anwendungen oder Ressourcen genutzt wird. Die Dienstvorlage definiert die Art der Infrastruktur, die für die Bereitstellung und Wartung einer einzelnen Anwendung oder eines einzelnen Microservices in einer Umgebung erforderlich ist. Ein AWS Proton Dienst ist eine Instanziierung einer Dienstvorlage, die normalerweise mehrere Dienstinstanzen und

eine Pipeline umfasst. Eine AWS Proton Dienstinstanz ist eine Instanziierung einer Dienstvorlage in einer bestimmten Umgebung. Sie oder andere Mitglieder Ihres Teams können angeben, welche Umgebungsvorlagen mit einer bestimmten Dienstvorlage kompatibel sind. Weitere Informationen zu Vorlagen finden Sie unterAWS Proton Vorlagen.

Sie können die folgende Infrastruktur als Codeanbieter verwenden mit AWS Proton:

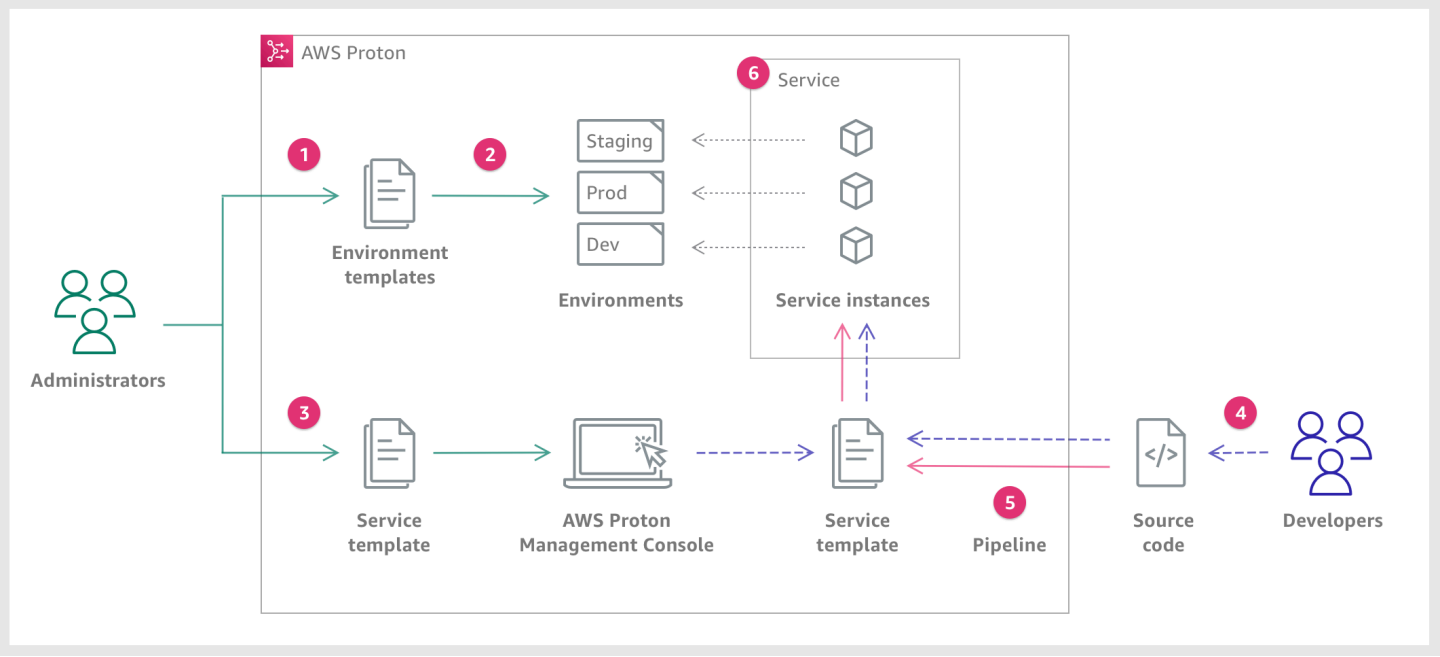
- CloudFormation
- Terraform

AWS Proton für Entwickler

Als Anwendungsentwickler wählen Sie eine standardisierte Dienstvorlage aus, AWS Proton mit der Sie einen Dienst erstellen, der Ihre Anwendung in einer Dienstinstanz bereitstellt und verwaltet. Ein AWS Proton Service ist eine Instanziierung einer Dienstvorlage, die normalerweise mehrere Dienstinstanzen und eine Pipeline umfasst.

AWS Proton Arbeitsablauf

Das folgende Diagramm ist eine Visualisierung der wichtigsten AWS Proton Konzepte, die im vorherigen Absatz erörtert wurden. Es bietet auch einen allgemeinen Überblick darüber, was einen einfachen AWS Proton Arbeitsablauf ausmacht.



1. Als Administrator erstellen und registrieren Sie eine Umgebungsvorlage AWS Proton, die die gemeinsam genutzten Ressourcen definiert.
2. AWS Proton stellt eine oder mehrere Umgebungen auf der Grundlage einer Umgebungsvorlage bereit.
3. Als Administrator erstellen und registrieren Sie eine Dienstvorlage AWS Proton, die die zugehörige Infrastruktur, Überwachung und CI/CD Ressourcen sowie kompatible Umgebungsvorlagen definiert.
4. Als Entwickler wählen Sie eine registrierte Service-Vorlage aus und geben einen Link zu Ihrem Quellcode-Repository an.
5. AWS Proton stellt den Service mit einer CI/CD-Pipeline für Ihre Service-Instanzen bereit.
6. AWS Proton stellt den Service und die Service-Instanzen bereit und verwaltet sie, auf denen der Quellcode ausgeführt wird, wie er in der ausgewählten Service-Vorlage definiert wurde. Eine Service-Instanz ist eine Instanziierung der ausgewählten Service-Vorlage in einer Umgebung für eine einzelne Phase einer Pipeline (zum Beispiel Prod).

AWS Proton Leitfaden für veraltete Dienste und Migration

AWS hat beschlossen, das Programm einzustellen AWS Proton, wobei der Support am 7. Oktober 2026 endet. Neukunden können sich nach dem 7. Oktober 2025 nicht mehr anmelden, Bestandskunden können den Service jedoch bis zum 7. Oktober 2026 weiter nutzen.

Servicestatus bis zur Deprecation

Bis zum 7. Oktober 2026 können AWS Proton Bestandskunden den Service weiterhin normal nutzen. Während dieses Zeitraums AWS wird:

1. Bereitstellung von Sicherheitspatches und kritischen Bugfixes

2. Sorgen Sie für Serviceverfügbarkeit und Leistung
3. Bieten Sie weiterhin Support über AWS Support Kanäle an
4. Fügen Sie dem Service keine neuen Funktionen hinzu

Wichtige Informationen zur Migration

AWS Proton ist in erster Linie ein CI/CD Tool für die Bereitstellung von Infrastruktur. Wenn veraltet AWS Proton ist, bleiben Ihre bereitgestellten CloudFormation Stacks und die Ressourcen, die sie verwalten, intakt und funktionieren weiterhin. Die Einstellung wirkt sich nur auf die Bereitstellungspipelines und den AWS Proton Service selbst aus, nicht auf Ihre bereitgestellte Infrastruktur.

Alternative Lösungen

Wir haben mehrere Alternativen identifiziert AWS Proton , die Ihnen helfen können, Ihre Infrastruktur in Form von Code und CI/CD-Funktionen aufrechtzuerhalten.

CloudFormation Git-Synchronisierung

Am besten geeignet für: Teams CloudFormation , die Who Want einen GitOps Workflow verwenden

Mit Git Sync können Plattformteams CloudFormation Vorlagen in einem Git-Repository modellieren, das Entwicklungsteams forken können. Entwickler aktualisieren Parameterdateien, übertragen Änderungen an ihr Fork-Repository und Git Sync aktualisiert den Stack.

Die wichtigsten Vorteile:

1. Ähnliche Entwicklererfahrung wie AWS Proton
2. Nutzt vorhandenes Wissen CloudFormation
3. Klare Trennung zwischen Plattform- und Entwicklerteams

Einschränkungen:

1. Kein Konzept von Umgebungen
2. Keine erweiterten Pipeline-Funktionen
3. Verlässt sich auf GitHub Funktionen, die bei anderen Git-Anbietern möglicherweise nicht verfügbar sind

Erfahren Sie mehr: Git Sync

Harmonix aktiviert AWS

Ideal für: Unternehmen, die ein umfassendes internes Entwicklerportal benötigen

Harmonix ist eine AWS Partner Lösung, die auf Backstage.io basiert und ein AWS Plugin bietet, mit dem Teams Vorlagen, Umgebungen und Dienste erstellen können, die Proton ähneln.

Die wichtigsten Vorteile:

1. Ähnliche Funktionalität wie AWS Proton
2. Basiert auf dem beliebten Backstage-Framework
3. Umfassendes Erlebnis im Entwicklerportal

Einschränkungen:

1. Wird nicht von einem AWS-Service Team verwaltet
2. Referenzimplementierung, die möglicherweise angepasst werden muss

Erfahren Sie mehr: https://harmonixonaws.io/

AWS CodePipeline und AWS CodeBuild

Am besten geeignet für: Teams, die maximale Flexibilität und Kontrolle benötigen

Verwenden Sie die AWS grundlegenden CI/CD Dienste, um AWS Proton Funktionen flexibler und kontrollierter zu replizieren.

Die wichtigsten Vorteile:

1. Maximale Flexibilität
2. Tiefe Integration mit AWS Diensten
3. Aktive Wartung und neue Funktionen

Einschränkungen:

1. Erfordert mehr Implementierungsarbeit

2. Weniger Self-Service für out-of-box Entwickler

Weitere Informationen:

Was ist AWS CodePipeline

Was ist AWS CodeBuild

GitHub Aktionen

Ideal für: Kleinere Teams GitHub , die Wert auf Einfachheit legen

>Die wichtigsten Vorteile

1. Einfache Integration mit Repositorien GitHub
2. Einfache Einrichtung für Benutzer GitHub
3. Großer Marktplatz für wiederverwendbare Aktionen

Einschränkungen:

1. An das GitHub Ökosystem gebunden
2. Erfordert möglicherweise mehr Arbeit für die Steuerung durch das Plattformteam

Weitere Informationen:

GitHub Dokumentation der Aktionen

CI/CD-Beispiel: Integration mit GitHub Aktionen — CI/CD Pipeline zur Bereitstellung einer Web-App für Amazon EC2

Anleitung zur Migration

Der Migrationsprozess hängt von Ihrer Implementierung und der gewählten Alternative ab. Allgemeine Schritte:

1. Inventarisieren Sie Ihre Proton-Ressourcen:
2. Wählen Sie eine alternative Lösung:
3. Extrahieren Sie Ihre Vorlagendaten:

4. Implementieren Sie die von Ihnen gewählte Alternative:
5. Migrieren Sie Produktionsworkloads:

Wenn Sie spezielle Unterstützung bei der Migration benötigen, AWS Support wenden Sie sich an Ihr Account-Team.

FAQs

F: Warum wird die AWS Einstellung eingestellt? AWS Proton A: Wir haben bessere Möglichkeiten identifiziert, die Kundenanforderungen im Bereich Infrastruktur als Code durch andere AWS Partner Lösungen AWS zu erfüllen.

F: Funktioniert meine bestehende Infrastruktur auch nach dem Verfallsdatum weiter? A: Ja. AWS Proton ist in erster Linie ein CI/CD Werkzeug. Ihre bereitgestellten CloudFormation Stacks und die Ressourcen, die sie verwalten, bleiben intakt und funktionieren weiterhin. Die veraltete Version wirkt sich nur auf die Bereitstellungspipelines aus, nicht auf Ihre bereitgestellte Infrastruktur.

F: Wie kann ich Hilfe bei der Migration erhalten? A: AWS Support kann Sie bei Ihrer Migration unterstützen. Bitte wenden Sie sich an Ihren Manager AWS Support, oder Sie können sich an Ihren AWS-Konto Manager wenden, um Unterstützung zu erhalten.

F: Welche Alternative sollte ich wählen? A: Die beste Alternative hängt von Ihrem spezifischen Anwendungsfall ab:

1. Für einen einfachen GitOps Workflow: CloudFormation Git Sync
2. Für Unternehmen, die ein Entwicklerportal benötigen: Harmonix On AWS
3. Für maximale Flexibilität: und AWS CodePipeline AWS CodeBuild
4. Für Teams, die bereits dabei sind GitHub: GitHub Aktionen

F: Was passiert, wenn ich nicht bis zum 7. Oktober 2026 migriere? A: Sie werden nicht mehr darauf zugreifen AWS Proton können. Ihre bestehende Infrastruktur wird weiterhin funktionieren, Sie können sie jedoch nicht verwenden, um sie AWS Proton zu verwalten oder zu aktualisieren.

F: Wie lange werden meine Daten aufbewahrt? A: Bis zum 7. Oktober 2026. Nach diesem Datum werden alle Daten gelöscht.

Wenn Sie weitere Fragen haben, wenden Sie sich bitte an AWS Support.

Einrichtung

Führen Sie die Aufgaben in diesem Abschnitt aus, damit Sie Service- und Umgebungsvorlagen erstellen und registrieren können. Sie benötigen diese, um Umgebungen und Dienste bereitzustellen AWS Proton.

> **Note**
>
> Wir bieten sie ohne zusätzliche Kosten AWS Proton an. Sie können Service- und Umgebungsvorlagen kostenlos erstellen, registrieren und verwalten. Sie können sich auch AWS Proton darauf verlassen, dass die eigenen Abläufe wie Speicher, Sicherheit und Bereitstellung selbst verwaltet werden. Die einzigen Kosten, die Ihnen bei der Nutzung entstehen, AWS Proton sind die folgenden.
>
> - Kosten für die Bereitstellung und Nutzung von AWS Cloud Ressourcen, die Sie in Auftrag gegeben haben AWS Proton , für Sie bereitzustellen und zu warten.
> - Kosten für die Aufrechterhaltung einer AWS CodeStar Verbindung zu Ihrem Code-Repository.
> - Kosten für die Wartung eines Amazon S3 S3-Buckets, wenn Sie einen Bucket verwenden, um Eingaben dafür bereitzustellen AWS Proton. Sie können diese Kosten vermeiden, wenn Sie auf die the section called “Konfigurationen für die Vorlagensynchronisierung” Verwendung von Git-Repositorys für Ihre the section called “Vorlagenpakete” umsteigen.

Themen

Einrichtung mit IAM

Wenn Sie sich für registrieren AWS, AWS-Konto ist Ihr automatisch für alle Dienste in angemeldet AWS, einschließlich AWS Proton. Ihnen werden nur die Dienste und Ressourcen in Rechnung gestellt, die Sie nutzen.

> **Note**
>
> Sie und Ihr Team, einschließlich Administratoren und Entwickler, müssen sich alle unter demselben Konto befinden.

Melden Sie sich an für AWS

Wenn Sie noch keine haben AWS-Konto, führen Sie die folgenden Schritte aus, um eine zu erstellen.

Um sich für eine anzumelden AWS-Konto

1. Öffnen Sie https://portal.aws.amazon.com/billing/die Anmeldung.
2. Folgen Sie den Online-Anweisungen.

   Während der Anmeldung erhalten Sie einen Telefonanruf oder eine Textnachricht und müssen einen Verifizierungscode über die Telefontasten eingeben.

   Wenn Sie sich für eine anmelden AWS-Konto, Root-Benutzer des AWS-Kontoswird eine erstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Als bewährte Sicherheitsmethode weisen Sie einem Administratorbenutzer Administratorzugriff zu und verwenden Sie nur den Root-Benutzer, um Aufgaben auszuführen, die Root-Benutzerzugriff erfordern.

Erstellen eines IAM-Benutzers

Wählen Sie zum Erstellen eines Administratorbenutzers eine der folgenden Optionen aus.

Wählen Sie eine Möglichkeit zur Verwaltung Ihres Administrators aus.	Bis	Von	Sie können auch
Im IAM Identity Center (Empfohlen)	Verwendung von kurzfristigen Anmeldeinformationen für den Zugriff auf AWS. Dies steht im Einklang mit den bewährten Methoden für die Sicherheit. Weitere Informationen zu bewährten Methoden finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.	Beachtung der Anweisungen unter Erste Schritte im AWS IAM Identity Center - Benutzerhandbuch.	Konfigurieren Sie den programmatischen Zugriff, indem Sie AWS CLI die Konfiguration für die Verwendung AWS IAM Identity Center im AWS Command Line Interface Benutzerhandbuch vornehmen.
In IAM (Nicht empfohlen)	Verwendung von langfristigen Anmeldeinformationen für den Zugriff auf AWS.	Folgen Sie den Anleitungen unter IAM-Benutzer für den Notfallzugriff erstellen im IAM-Benutzerhandbuch.	Sie konfigurieren den programmgesteuerten Zugriff unter Verwendung der Informationen unter Verwalten der Zugriffsschlüssel für IAM-Benutzer im IAM-Benutzerhandbuch.

AWS Proton Servicerollen einrichten

Es gibt einige IAM-Rollen, die Sie möglicherweise für verschiedene Teile Ihrer AWS Proton Lösung erstellen möchten. Sie können sie im Voraus mit der IAM-Konsole erstellen, oder Sie können die AWS Proton Konsole verwenden, um sie für Sie zu erstellen.

Erstellen Sie AWS Proton Umgebungsrollen, AWS Proton damit Sie in Ihrem Namen API-Aufrufe an andere AWS-Services CloudFormation AWS CodeBuild, z. B. verschiedene Rechen- und Speicherdienste tätigen können, um Ressourcen für Sie bereitzustellen. Eine AWS-verwaltete Bereitstellungsrolle ist erforderlich, wenn eine Umgebung oder eine der darin ausgeführten Dienstinstanzen AWS-managed Provisioning verwendet. Eine CodeBuildRolle ist erforderlich, wenn eine Umgebung oder eine ihrer Dienstinstanzen Provisioning verwendet. CodeBuild Weitere Informationen zu den AWS Proton Umgebungsrollen finden Sie unterthe section called “IAM-Rollen”. Wenn Sie eine Umgebung erstellen, können Sie die AWS Proton Konsole verwenden, um eine vorhandene Rolle für eine dieser beiden Rollen auszuwählen oder eine Rolle mit Administratorrechten für Sie zu erstellen.

Erstellen Sie auf ähnliche Weise AWS Proton Pipeline-Rollen, AWS Proton um API-Aufrufe an andere Dienste in Ihrem Namen zu tätigen, um eine CI/CD-Pipeline für Sie bereitzustellen. Weitere Informationen zu den AWS Proton Pipeline-Rollen finden Sie unter. the section called “Rollen im Pipeline-Dienst” Weitere Informationen zur Konfiguration von CI/CD Einstellungen finden Sie unterthe section called “Einstellungen für die CI/CD Konto-Pipeline einrichten”.

> **Note**
>
> Da wir nicht wissen, welche Ressourcen Sie in Ihren AWS Proton Vorlagen definieren werden, verfügen die Rollen, die Sie mit der Konsole erstellen, über umfassende Berechtigungen und können sowohl als AWS Proton Pipeline-Dienstrollen als auch als Servicerollen verwendet werden. AWS Proton Für Produktionsbereitstellungen empfehlen wir, die Berechtigungen auf die spezifischen Ressourcen zu beschränken, die bereitgestellt werden, indem Sie benutzerdefinierte Richtlinien sowohl für die AWS Proton Pipeline-Dienstrollen als auch für die AWS Proton Umgebungsdienstrollen erstellen. Sie können diese Rollen mithilfe von AWS CLI oder IAM erstellen und anpassen. Weitere Informationen erhalten Sie unter Servicerollen für AWS Proton und Einen Service erstellen.

Einrichtung mit AWS Proton

Wenn Sie den AWS CLI zum Ausführen verwenden möchten, stellen Sie sicher AWS Proton APIs, dass Sie ihn installiert haben. Wenn Sie es nicht installiert haben, finden Sie weitere Informationen unterEinrichtung der AWS CLI.

AWS Proton spezifische Konfiguration:

- Um Vorlagen zu erstellen und zu verwalten:
  - Wenn Sie Konfigurationen für die Vorlagensynchronisierung verwenden, richten Sie eine AWS CodeStar Verbindung ein.
  - Andernfalls richten Sie einen Amazon S3 S3-Bucket ein.
- So stellen Sie die Infrastruktur bereit:
  - Für die selbstverwaltete Bereitstellung müssen Sie eine AWS CodeStar Verbindung einrichten.
- (Optional) So stellen Sie Pipelines bereit:
  - Richten Sie für AWS-managed Provisioning und CodeBuildbased Provisioning Pipeline-Rollen ein.
  - Richten Sie für die selbstverwaltete Bereitstellung ein Pipeline-Repository ein.

Weitere Informationen zu Bereitstellungsmethoden finden Sie unter. the section called “AWS-verwaltete Bereitstellung”

Einen Amazon S3 S3-Bucket einrichten

Um einen S3-Bucket einzurichten, folgen Sie den Anweisungen unter Erstellen Sie Ihren ersten S3-Bucket, um einen S3-Bucket einzurichten. Platzieren Sie Ihre Eingaben AWS Proton in dem Bucket, wo AWS Proton Sie sie abrufen können. Diese Eingaben werden als Template-Bundles bezeichnet. In anderen Abschnitten dieses Handbuchs erfahren Sie mehr über sie.

Eine AWS CodeStar Verbindung einrichten

Um eine Verbindung AWS Proton zu einem Repository herzustellen, erstellen Sie eine AWS CodeStar Verbindung, die eine Pipeline aktiviert, wenn ein neuer Commit in einem Quellcode-Repository eines Drittanbieters vorgenommen wird.

AWS Proton verwendet die Verbindung für:

- Aktiviert eine Service-Pipeline, wenn ein neuer Commit für den Quellcode Ihres Repositorys vorgenommen wird.
- Stellen Sie eine Pull-Anfrage für ein Infrastruktur-as-Code-Repository.
- Erstellen Sie immer dann eine neue Neben- oder Hauptversion einer Vorlage, wenn ein Commit in ein Template-Repository übertragen wird, das eine Ihrer Vorlagen ändert, sofern die Version noch nicht existiert.

Du kannst dich mit Bitbucket- GitHub, GitHub Enterprise- und GitHub Enterprise Server-Repositorys verbinden. CodeConnections Weitere Informationen finden Sie unter CodeConnections im AWS CodePipeline -Benutzerhandbuch.

Um eine CodeStar Verbindung einzurichten.

1. Öffnen Sie die AWS Proton -Konsole.
2. Wählen Sie im Navigationsbereich Einstellungen und dann Repository-Verbindungen aus, um zur Seite Verbindungen in den Einstellungen der Developer Tools zu gelangen. Auf der Seite wird eine Liste von Verbindungen angezeigt.
3. Wählen Sie Verbindung erstellen und folgen Sie den Anweisungen.

Einstellungen für die CI/CD Konto-Pipeline einrichten

AWS Proton kann CI/CD Pipelines für die Bereitstellung von Anwendungscode in Ihren Serviceinstanzen bereitstellen. Die AWS Proton Einstellungen, die Sie für die Pipeline-Bereitstellung benötigen, hängen von der Bereitstellungsmethode ab, die Sie für Ihre Pipeline wählen.

AWS-verwaltete und CodeBuild basierte Bereitstellung — Richten Sie Pipeline-Rollen ein

Mit AWS-managed Provisioning und CodeBuild Provisioning werden Pipelines für Sie bereitgestellt. AWS Proton Benötigt daher eine AWS Proton Servicerolle, die Berechtigungen für die Bereitstellung von Pipelines bereitstellt. Jede dieser beiden Bereitstellungsmethoden verwendet ihre eigene Servicerolle. Diese Rollen werden von allen AWS Proton Service-Pipelines gemeinsam genutzt und Sie konfigurieren sie einmal in Ihren Kontoeinstellungen.

So erstellen Sie Pipeline-Servicerollen mithilfe der Konsole

1. Öffnen Sie die AWS Proton -Konsole.
2. Wählen Sie im Navigationsbereich Einstellungen und dann Kontoeinstellungen aus.
3. Wählen Sie auf der Seite mit den CI/CD Kontoeinstellungen die Option Konfigurieren aus.
4. Führen Sie eine der folgenden Aktionen aus:
   - Um eine Pipeline-Servicerolle für Sie AWS Proton erstellen zu lassen

     [Um die AWS-verwaltete Bereitstellung von Pipelines zu aktivieren] Gehen Sie auf der Seite Kontoeinstellungen konfigurieren im Abschnitt AWS-managed provisioning Pipeline-Rolle wie folgt vor:

     a. Wählen Sie Neue Servicerolle aus.
     b. Geben Sie einen Namen für die Rolle ein, zum Beispiel**`myProtonPipelineServiceRole`**.
     c. Aktivieren Sie das Kontrollkästchen, um der Erstellung einer AWS Proton Rolle mit Administratorrechten in Ihrem Konto zuzustimmen.

     [Um die CodeBuild basierte Bereitstellung von Pipelines zu aktivieren] Wählen Sie auf der Seite Kontoeinstellungen konfigurieren im Abschnitt CodeBuildPipeline-Rolle die Option Bestehende Servicerolle und wählen Sie die Servicerolle aus, die Sie im Abschnitt CloudFormation Pipeline-Rolle erstellt haben. Oder, wenn Sie keine CloudFormation Pipeline-Rolle zugewiesen haben, wiederholen Sie die vorherigen drei Schritte, um eine neue Servicerolle zu erstellen.
   - Um bestehende Pipeline-Dienstrollen auszuwählen

     [Um die AWS verwaltete Bereitstellung von Pipelines zu aktivieren] Wählen Sie auf der Seite Kontoeinstellungen konfigurieren im Abschnitt AWS-managed provisioning pipeline role die Option Existierende Servicerolle und wählen Sie eine Servicerolle in Ihrem Konto aus. AWS

     [Um die CodeBuild Bereitstellung von Pipelines zu aktivieren] Wählen Sie auf der Seite Kontoeinstellungen konfigurieren im Abschnitt CodeBuildPipeline-Bereitstellungsrolle die Option Bestehende Servicerolle und wählen Sie eine Servicerolle in Ihrem Konto aus. AWS
5. Wählen Sie Änderungen speichern aus.

   Ihre neue Pipeline-Servicerolle wird auf der Seite mit den Kontoeinstellungen angezeigt.

Selbstverwaltete Bereitstellung — Richten Sie ein Pipeline-Repository ein

AWS Proton Sendet bei der selbstverwalteten Bereitstellung eine Pull-Anfrage (PR) an ein von Ihnen eingerichtetes Provisioning-Repository, und Ihr Automatisierungscode ist für die Bereitstellung von Pipelines verantwortlich. Für die Bereitstellung von Pipelines ist daher AWS Proton keine Servicerolle erforderlich. Stattdessen benötigt es ein registriertes Provisioning-Repository. Ihr Automatisierungscode im Repository muss eine entsprechende Rolle übernehmen, die Berechtigungen für die Bereitstellung von Pipelines bereitstellt.

Um ein Pipeline-Provisioning-Repository mit der Konsole zu registrieren

1. Erstellen Sie ein CI/CD Pipeline-Provisioning-Repository, falls Sie noch keines erstellt haben. Weitere Informationen zu Pipelines bei der selbstverwalteten Bereitstellung finden Sie unter. the section called “Selbstverwaltete Bereitstellung”
2. Wählen Sie im Navigationsbereich Einstellungen und dann Kontoeinstellungen aus.
3. Wählen Sie auf der Seite mit den CI/CD Kontoeinstellungen die Option Konfigurieren aus.
4. Gehen Sie auf der Seite Kontoeinstellungen konfigurieren im Abschnitt CI/CD-Pipeline-Repository wie folgt vor:

   a. Wählen Sie Neues Repository und dann einen der Repository-Anbieter aus.

   b. Wählen Sie für die CodeStar Verbindung eine Ihrer Verbindungen aus.

   > **Note**
   >
   > Wenn Sie noch keine Verbindung zum entsprechenden Repository-Provider-Konto haben, wählen Sie Neue CodeStar Verbindung hinzufügen, schließen Sie den Verbindungsaufbau ab und klicken Sie dann auf die Schaltfläche „Aktualisieren“ neben dem CodeStarVerbindungsmenü. Sie sollten jetzt in der Lage sein, Ihre neue Verbindung im Menü auszuwählen.

   c. Wählen Sie als Repository-Name Ihr Pipeline-Provisioning-Repository aus. Das Dropdownmenü zeigt die Liste der Repositorys im Anbieterkonto.

   d. Wählen Sie als Branch-Name einen der Repository-Branches aus.
5. Wählen Sie Änderungen speichern aus.

   Ihr Pipeline-Repository wird auf der Seite mit den Kontoeinstellungen angezeigt.

Einrichtung der AWS CLI

Um die AWS CLI für AWS Proton API-Aufrufe zu verwenden, stellen Sie sicher, dass Sie die neueste Version von installiert haben AWS CLI. Weitere Informationen finden Sie unter Erste Schritte mit dem AWS CLI im AWS Command Line Interface -Benutzerhandbuch. Weitere Informationen zu den ersten Schritten AWS CLI mit AWS Proton finden Sie unterthe section called “Erste Schritte mit der CLI”.

Erste Schritte mit AWS Proton

Bevor Sie beginnen, sollten Sie sich für die Nutzung einrichtenAWS Proton und überprüfen, ob Sie die Voraussetzungen für die ersten Schritte erfüllt haben.

Wählen Sie AWS Proton zunächst einen oder mehrere der folgenden Pfade aus:

- Folgen Sie einem angeleiteten Beispielkonsolen- oder CLI-Workflow über Dokumentationslinks.
- Führen Sie einen angeleiteten Beispiel-Konsolen-Workflow durch.
- Führen Sie einen AWS CLI Beispiel-Workflow mit Anleitung durch.

Topics

Voraussetzungen

Bevor Sie mit der Verwendung beginnen AWS Proton, stellen Sie sicher, dass die folgenden Voraussetzungen erfüllt sind. Weitere Informationen finden Sie unter Einrichtung.

- Sie haben ein IAM-Konto mit Administratorrechten. Weitere Informationen finden Sie unter Einrichtung mit IAM.
- Sie haben die AWS Proton Servicerolle und die AWS Proton Pipeline-Servicerolle sind mit Ihrem Konto verknüpft. Weitere Informationen erhalten Sie unter AWS Proton Servicerollen einrichten und Servicerollen für AWS Proton.
- Sie haben eine AWS CodeStar Verbindung. Weitere Informationen finden Sie unter Eine AWS CodeStar Verbindung einrichten.
- Sie sind mit der Erstellung von CloudFormation Vorlagen und der Jinja-Parametrisierung vertraut. Weitere Informationen finden Sie unter Was ist? CloudFormation im CloudFormation Benutzerhandbuch und auf der Jinja-Website.

- Sie verfügen über praktische Kenntnisse im Bereich AWS Infrastrukturdienste.
- Sie sind bei Ihrem angemeldet AWS-Konto.

Arbeitsablauf „Erste Schritte“

Erfahren Sie anhand der Beispielschritte und Links, wie Sie Vorlagenpakete erstellen, Vorlagen erstellen und registrieren und Umgebungen und Dienste erstellen.

Stellen Sie vor dem Start sicher, dass Sie eine AWS Proton Servicerolle erstellt haben.

Wenn Ihre Dienstvorlage eine AWS Proton Dienstpipeline enthält, stellen Sie sicher, dass Sie eine AWS CodeStar Verbindung und eine AWS Proton Pipeline-Servicerolle erstellt haben.

Weitere Informationen finden Sie unter Die AWS Proton Service-API-Referenz.

Beispiel: Workflow „Erste Schritte“

1. Eine allgemeine Übersicht der AWS Proton Eingaben und Ausgaben finden Sie im Wie AWS Proton funktioniert Diagramm unter.
2. Erstellen Sie ein Umgebungs- und ein Dienstvorlagenpaket.
    a. Identifizieren Sie die Eingabeparameter.
    b. Erstellen Sie eine Schemadatei.
    c. Erstellen Sie Infrastructure-as-Code-Dateien (IaC).
    d. Um Ihr Vorlagenpaket zusammenzufassen, erstellen Sie eine Manifestdatei und organisieren Sie Ihre IaC-Dateien, Manifestdateien und Schemadateien in Verzeichnissen.
    e. Machen Sie Ihr Vorlagenpaket zugänglich AWS Proton für.
3. Erstellen und registrieren Sie eine Umgebungsvorlagenversion mit AWS Proton.

    Wenn Sie die Konsole verwenden, um eine Vorlage zu erstellen und zu registrieren, wird automatisch eine Vorlagenversion erstellt.

    Wenn Sie die verwenden AWS CLI , um eine Vorlage zu erstellen und zu registrieren:

    a. Erstellen Sie eine Umgebungsvorlage.
    b. Erstellen Sie eine Version der Umgebungsvorlage.

Weitere Informationen finden Sie unter CreateEnvironmentTemplateund CreateEnvironmentTemplateVersionin der AWS Proton API-Referenz.

4. Veröffentlichen Sie Ihre Umgebungsvorlage, um sie für die Verwendung verfügbar zu machen.

   Weitere Informationen finden Sie UpdateEnvironmentTemplateVersionin der AWS Proton API-Referenz.

5. Um eine Umgebung zu erstellen, wählen Sie eine veröffentlichte Version der Umgebungsvorlage aus und geben Sie Werte für die erforderlichen Eingaben an.

   Weitere Informationen finden Sie CreateEnvironmentin der AWS Proton API-Referenz.

6. Erstellen und registrieren Sie eine Service-Vorlagenversion mit AWS Proton.

   Wenn Sie die Konsole verwenden, um eine Vorlage zu erstellen und zu registrieren, wird automatisch eine Vorlagenversion erstellt.

   Wenn Sie die verwenden AWS CLI , um eine Vorlage zu erstellen und zu registrieren:

   a. Erstellen Sie eine Dienstvorlage.
   b. Erstellen Sie eine Version der Dienstvorlage.

   Weitere Informationen finden Sie unter CreateServiceTemplateund CreateServiceTemplateVersionin der AWS Proton API-Referenz.

7. Veröffentlichen Sie Ihre Dienstvorlage, um sie für die Verwendung verfügbar zu machen.

   Weitere Informationen finden Sie UpdateServiceTemplateVersionin der AWS Proton API-Referenz.

8. Um einen Service zu erstellen, wählen Sie eine veröffentlichte Service-Vorlagenversion aus und geben Sie Werte für die erforderlichen Eingaben an.

   Weitere Informationen finden Sie CreateServicein der AWS Proton API-Referenz.

Erste Schritte mit dem AWS-Managementkonsole

Fangen Sie an mit AWS Proton

- Erstellen Sie eine Umgebungsvorlage und zeigen Sie sie an.

- Erstellen, Anzeigen und Veröffentlichen einer Dienstvorlage, die die Umgebungsvorlage verwendet, die Sie gerade erstellt haben.
- Erstellen Sie eine Umgebung und einen Dienst (optional).
- Löschen Sie die Dienstvorlage, die Umgebungsvorlage, die Umgebung und den Dienst, falls sie erstellt wurden.

Schritt 1: Öffnen Sie die AWS Proton Konsole

- Öffnen Sie die AWS Proton -Konsole.

Schritt 2: Bereiten Sie sich auf die Verwendung der Beispielvorlagen vor

1. Erstellen Sie eine Codestar-Verbindung zu Github und benennen Sie die Verbindung. my-proton-connection
2. Navigieren Sie zu https://github.com/aws-samples/aws-proton-cloudformation-sample-templates
3. Erstelle einen Fork des Repositorys in deinem Github-Konto.

Schritt 3: Erstellen Sie eine Umgebungsvorlage

Wählen Sie im Navigationsbereich die Option Umgebungsvorlagen aus.

1. Wählen Sie auf der Seite Umgebungsvorlagen die Option Umgebungsvorlage erstellen aus.
2. Wählen Sie auf der Seite Umgebungsvorlage erstellen im Abschnitt Vorlagenoptionen die Option Vorlage für die Bereitstellung neuer Umgebungen erstellen aus.
3. Wählen Sie im Abschnitt Quelle des Vorlagenpakets die Option Ein Vorlagenpaket aus Git synchronisieren aus.
4. Wählen Sie im Abschnitt Repository für Vorlagendefinitionen die Option Ein verknüpftes Git-Repository auswählen aus.
5. Wählen Sie my-proton-connectionaus der Repository-Liste aus.
6. Wählen Sie main aus der Branch-Liste aus.
7. Im Abschnitt Details zur Proton-Umgebungsvorlage.
    a. Geben Sie den Vorlagennamen als **`fargate-env`** ein.
    b. Geben Sie den Anzeigenamen der Umgebungsvorlage als ein**`My Fargate Environment`**.

c. (Optional) Geben Sie eine Beschreibung für die Umgebungsvorlage ein.

8. (Optional) Wählen Sie im Abschnitt „Tags“ die Option Neues Tag hinzufügen aus und geben Sie einen Schlüssel und einen Wert ein, um ein vom Kunden verwaltetes Tag zu erstellen.
9. Wählen Sie „Umgebungsvorlage erstellen“.

   Sie befinden sich jetzt auf einer neuen Seite, auf der der Status und die Details für Ihre neue Umgebungsvorlage angezeigt werden. Zu diesen Details gehören eine Liste von AWS und vom Kunden verwaltete Tags. AWS Proton generiert automatisch AWS verwaltete Tags für Sie, wenn Sie AWS Proton Ressourcen erstellen. Weitere Informationen finden Sie unter AWS Proton Ressourcen und Tagging.
10. Der Status einer neuen Umgebungsvorlage beginnt im Status Entwurf. Sie und andere Personen mit `proton:CreateEnvironment` entsprechenden Berechtigungen können die Datei anzeigen und darauf zugreifen. Folgen Sie dem nächsten Schritt, um die Vorlage für andere verfügbar zu machen.
11. Wählen Sie im Abschnitt Vorlagenversionen das Optionsfeld links neben der Nebenversion der Vorlage, die Sie gerade erstellt haben (1.0). Alternativ können Sie im Info-Banner die Option Veröffentlichen auswählen und den nächsten Schritt überspringen.
12. Wählen Sie im Abschnitt Vorlagenversionen die Option Veröffentlichen aus.
13. Der Status der Vorlage ändert sich in Veröffentlicht. Da es sich um die neueste Version der Vorlage handelt, handelt es sich um die empfohlene Version.
14. Wählen Sie im Navigationsbereich die Option Umgebungsvorlagen aus.

    Auf einer neuen Seite wird eine Liste Ihrer Umgebungsvorlagen zusammen mit Vorlagendetails angezeigt.

Schritt 4: Erstellen Sie eine Dienstvorlage

Erstellen Sie eine Dienstvorlage.

1. Wählen Sie im Navigationsbereich die Option Dienstvorlagen aus.
2. Wählen Sie auf der Seite mit den Dienstvorlagen die Option Dienstvorlage erstellen aus.
3. Wählen Sie auf der Seite Dienstvorlage erstellen im Abschnitt Quelle des Vorlagenpakets die Option Vorlagenpaket aus Git synchronisieren aus.
4. Wählen Sie im Abschnitt Vorlage die Option Ein verknüpftes Git-Repository auswählen aus.
5. Wählen Sie my-proton-connectionaus der Repository-Liste aus.

6. Wählen Sie main aus der Branch-Liste aus.
7. Im Abschnitt Details zur Proton-Servicevorlage.
   a. Geben Sie den Namen der Dienstvorlage als **`backend-fargate-svc`** ein.
   b. Geben Sie den Anzeigenamen der Dienstvorlage als ein**`My Fargate Service`**.
   c. (Optional) Geben Sie eine Beschreibung für die Dienstvorlage ein.
8. Im Abschnitt Kompatible Umgebungsvorlagen.
   - Aktivieren Sie das Kontrollkästchen links neben der Umgebungsvorlage My Fargate Environment, um die kompatible Umgebungsvorlage für die neue Dienstvorlage auszuwählen.
9. Behalten Sie für die Verschlüsselungseinstellungen die Standardeinstellungen bei.
10. Im Abschnitt Pipeline-Definition.
    - Lassen Sie die Schaltfläche Diese Vorlage enthält eine CI/CD Pipeline ausgewählt.
11. Wählen Sie Dienstvorlage erstellen aus.

    Sie befinden sich jetzt auf einer neuen Seite, auf der der Status und die Details für Ihre neue Servicevorlage angezeigt werden, einschließlich einer Liste von AWS und vom Kunden verwalteten Tags.
12. Der Status einer neuen Servicevorlage beginnt im Status Entwurf. Nur Administratoren können sie anzeigen und darauf zugreifen. Gehen Sie wie folgt vor, um die Dienstvorlage Entwicklern zur Verfügung zu stellen.
13. Wählen Sie im Abschnitt Vorlagenversionen das Optionsfeld links neben der Nebenversion der Vorlage, die Sie gerade erstellt haben (1.0). Alternativ können Sie im Info-Banner die Option Veröffentlichen auswählen und den nächsten Schritt überspringen.
14. Wählen Sie im Abschnitt Vorlagenversionen die Option Veröffentlichen aus.
15. Der Status der Vorlage ändert sich in Veröffentlicht.

    Die erste Nebenversion Ihrer Service-Vorlage ist veröffentlicht und steht Entwicklern zur Verwendung zur Verfügung. Da es sich um die neueste Version der Vorlage handelt, handelt es sich um die empfohlene Version.
16. Wählen Sie im Navigationsbereich die Option Dienstvorlagen aus.

    Auf einer neuen Seite wird eine Liste Ihrer Dienstvorlagen und Details angezeigt.

Schritt 5: Erstellen Sie eine Umgebung

Wählen Sie im Navigationsbereich Environments (Umgebungen) aus.

1. Wählen Sie Create environment (Umgebung erstellen) aus.
2. Wählen Sie auf der Seite Umgebungsvorlage auswählen die Vorlage aus, die Sie gerade erstellt haben. Es heißt My Fargate Environment. Wählen Sie dann Configure.
3. Wählen Sie auf der Seite Umgebung konfigurieren im Abschnitt Provisioning die Option Bereitstellung durch AWS Proton aus.
4. Wählen Sie im Abschnitt Bereitstellungskonto die Option Dies AWS-Konto aus.
5. Geben Sie unter Umgebungseinstellungen den Namen der Umgebung als ein**`my-fargate-environment`**.
6. Wählen Sie im Abschnitt Umgebungsrollen die Option Neue Servicerolle oder, falls Sie bereits eine AWS Proton Servicerolle erstellt haben, die Option Bestehende Servicerolle aus.

   a. Wählen Sie Neue Servicerolle aus, um eine neue Rolle zu erstellen.

      i. Geben Sie den Namen der Umgebungsrolle als ein**`MyProtonServiceRole`**.

      ii. Aktivieren Sie das Kontrollkästchen, um der Erstellung einer AWS Proton Servicerolle mit Administratorrechten für Ihr Konto zuzustimmen.

   b. Wählen Sie Bestehende Servicerolle aus, um eine bestehende Rolle zu verwenden.

      - Wählen Sie Ihre Rolle im Dropdownfeld Name der Umgebungsrolle aus.
7. Wählen Sie Weiter aus.
8. Verwenden Sie auf der Seite Benutzerdefinierte Einstellungen konfigurieren die Standardeinstellungen.
9. Wählen Sie Weiter und überprüfen Sie Ihre Eingaben.
10. Wählen Sie Erstellen aus.

    Sehen Sie sich die Umgebungsdetails und den Status sowie die AWS verwalteten und vom Kunden verwalteten Tags für Ihre Umgebung an.
11. Wählen Sie im Navigationsbereich Environments (Umgebungen) aus.

    Auf einer neuen Seite wird eine Liste Ihrer Umgebungen zusammen mit dem Status und anderen Umgebungsdetails angezeigt.

Schritt 6: Optional — Erstellen Sie einen Dienst und stellen Sie eine Anwendung bereit

1. Öffnen Sie die AWS Proton -Konsole.
2. Wählen Sie im Navigationsbereich Services.
3. Wählen Sie auf der Seite Dienste die Option Dienst erstellen aus.
4. Wählen Sie auf der Seite Servicevorlage auswählen die Vorlage My Fargate Service aus, indem Sie auf das Optionsfeld in der oberen rechten Ecke der Vorlagenkarte klicken.
5. Wählen Sie in der unteren rechten Ecke der Seite die Option Konfigurieren aus.
6. Geben Sie auf der Seite Dienst konfigurieren im Abschnitt Diensteinstellungen den Dienstnamen ein**`my-service`**.
7. (Optional) Geben Sie eine Beschreibung für den Dienst ein.
8. Gehen Sie im Abschnitt Einstellungen für das Service-Repository wie folgt vor:
   a. Wählen Sie für die CodeStar Verbindung Ihre Verbindung aus der Liste aus.
   b. Wählen Sie unter Repository-Name den Namen Ihres Quellcode-Repositorys aus der Liste aus.
   c. Wählen Sie unter Branch-Name den Namen Ihres Quellcode-Repository-Branches aus der Liste aus.
9. (Optional) Wählen Sie im Abschnitt „Tags“ die Option Neues Tag hinzufügen aus und geben Sie einen Schlüssel und einen Wert ein, um ein vom Kunden verwaltetes Tag zu erstellen. Klicken Sie anschließend auf Weiter.
10. Folgen Sie auf der Seite Benutzerdefinierte Einstellungen konfigurieren im Abschnitt Serviceinstanzen im Abschnitt Neue Instanz den nächsten Schritten, um benutzerdefinierte Werte für Ihre Service-Instance-Parameter anzugeben.
    a. Geben Sie den Instanznamen ein**`my-app-service`**.
    b. Wählen Sie die Umgebung **`my-fargate-environment`** für Ihre Dienstinstanz aus.
    c. Behalten Sie die Standardeinstellungen für die verbleibenden Instanzparameter bei.
    d. Behalten Sie die Standardeinstellungen für Pipeline-Eingaben bei.
    e. Wählen Sie Weiter und überprüfen Sie Ihre Eingaben.
    f. Wählen Sie Erstellen und sehen Sie sich Ihren Servicestatus und Ihre Servicedetails an.

11. Auf der Seite mit den Servicedetails können Sie den Status Ihrer Serviceinstanz und Pipeline einsehen, indem Sie die Registerkarten Übersicht und Pipeline auswählen. Auf diesen Seiten können Sie auch Tags einsehen AWS und vom Kunden verwalten. AWS Proton erstellt automatisch AWS verwaltete Tags für Sie. Wählen Sie Tags verwalten, um vom Kunden verwaltete Tags zu erstellen und zu ändern. Weitere Informationen über das Markieren mit Tags finden Sie unter AWS Proton Ressourcen und Tagging.
12. Wenn der Service aktiv ist, wählen Sie auf der Registerkarte Overview im Abschnitt Service Instances den Namen Ihrer Dienstinstanz aus my-app-service.

    Sie befinden sich jetzt auf der Detailseite der Service-Instanz.
13. Um Ihre Anwendung anzuzeigen, kopieren Sie im Abschnitt Ausgaben den ServiceEndpointLink in Ihren Browser.

    Sie sehen eine AWS Proton Grafik auf der Webseite.
14. Nachdem der Dienst erstellt wurde, wählen Sie im Navigationsbereich Dienste aus, um eine Liste Ihrer Dienste anzuzeigen.

Schritt 7: Aufräumen.

1. Öffnen Sie die AWS Proton -Konsole.
2. Löschen Sie einen Dienst (falls Sie einen erstellt haben)
    a. Wählen Sie im Navigationsbereich Services.
    b. Wählen Sie auf der Seite Dienste den Dienstnamen my-service aus.

       Sie befinden sich jetzt auf der Seite mit den Servicedetails für my-service.
    c. Wählen Sie in der oberen rechten Ecke der Seite Aktionen und dann Löschen aus.
    d. In einem Modal werden Sie aufgefordert, die Löschaktion zu bestätigen.
    e. Folgen Sie den Anweisungen und wählen Sie Ja, löschen.
3. Lösche eine Umgebung
    a. Wählen Sie im Navigationsbereich Environments (Umgebungen) aus.
    b. Wählen Sie auf der Seite Umgebungen das Optionsfeld links neben der Umgebung aus, die Sie gerade erstellt haben.
    c. Wählen Sie Aktionen und dann Löschen aus.

d. In einem Modalfenster werden Sie aufgefordert, die Löschaktion zu bestätigen.

e. Folgen Sie den Anweisungen und wählen Sie Ja, löschen.

4. Löschen Sie eine Dienstvorlage

   a. Wählen Sie im Navigationsbereich die Option Dienstvorlagen aus.

   b. Wählen Sie auf der Seite mit den Dienstvorlagen das Optionsfeld links neben der Dienstvorlage aus my-svc-template.

   c. Wählen Sie Aktionen und dann Löschen aus.

   d. In einem Modalfenster werden Sie aufgefordert, die Löschaktion zu bestätigen.

   e. Folgen Sie den Anweisungen und wählen Sie Ja, löschen. Dadurch werden die Dienstvorlage und alle ihre Versionen gelöscht.

5. Löscht eine Umgebungsvorlage

   a. Wählen Sie im Navigationsbereich die Option Umgebungsvorlagen aus.

   b. Wählen Sie auf der Seite mit den Umgebungsvorlagen das Optionsfeld links neben aus my-env-template.

   c. Wählen Sie Aktionen und dann Löschen aus.

   d. In einem Modalfenster werden Sie aufgefordert, die Löschaktion zu bestätigen.

   e. Folgen Sie den Anweisungen und wählen Sie Ja, löschen. Dadurch werden die Umgebungsvorlage und alle ihre Versionen gelöscht.

6. Löschen Sie Ihre Codestar-Verbindung

Erste Schritte mit dem AWS CLI

Folgen Sie diesem Tutorial AWS CLI, um mit der AWS Proton Verwendung von zu beginnen. Das Tutorial zeigt einen öffentlich zugänglichen AWS Proton Dienst mit Lastenausgleich auf der AWS Fargate Grundlage von. Das Tutorial stellt auch eine CI/CD Pipeline bereit, die eine statische Website mit einem angezeigten Bild bereitstellt.

Bevor Sie beginnen, stellen Sie sicher, dass Sie richtig eingerichtet sind. Details hierzu finden Sie unter the section called “Voraussetzungen”.

Schritt 1: Registrieren Sie eine Umgebungsvorlage

In diesem Schritt registrieren Sie als Administrator eine Beispielumgebungsvorlage, die einen Amazon Elastic Container Service (Amazon ECS) -Cluster und eine Amazon Virtual Private Cloud (Amazon VPC) mit zwei public/private Subnetzen enthält.

Um eine Umgebungsvorlage zu registrieren

1. Ordnen Sie das AWS Proton Sample CloudFormation Templates-Repository Ihrem GitHub Konto oder Ihrer Organisation zu. Dieses Repository enthält die Umgebungs- und Dienstvorlagen, die wir in diesem Tutorial verwenden.

   Registrieren Sie dann Ihr geforktes Repository bei AWS Proton. Weitere Informationen finden Sie unter the section called “Einen Repository-Link erstellen”.
2. Erstellen Sie eine Umgebungsvorlage.

   Die Umgebungsvorlagenressource verfolgt die Versionen der Umgebungsvorlagen.

   ```
   $ aws proton create-environment-template \
     --name "fargate-env" \
     --display-name "Public VPC Fargate" \
     --description "VPC with public access and ECS cluster"
   ```
3. Erstellen Sie eine Konfiguration für die Vorlagensynchronisierung.

   AWS Proton richtet eine Synchronisierungsbeziehung zwischen Ihrem Repository und Ihrer Umgebungsvorlage ein. Anschließend wird die Vorlagenversion 1.0 im DRAFT Status erstellt.

   ```
   $ 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. Warten Sie, bis die Version der Umgebungsvorlage erfolgreich registriert wurde.

   Wenn dieser Befehl mit dem Exit-Status von zurückkehrt0, ist die Versionsregistrierung abgeschlossen. Dies ist in Skripten nützlich, um sicherzustellen, dass Sie den Befehl im nächsten Schritt erfolgreich ausführen können.

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

5. Veröffentlichen Sie die Version der Umgebungsvorlage, um sie für die Erstellung der Umgebung verfügbar zu machen.

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

Schritt 2: Registrieren Sie eine Dienstvorlage

In diesem Schritt registrieren Sie als Administrator eine Beispiel-Servicevorlage, die alle Ressourcen enthält, die für die Bereitstellung eines Amazon ECS Fargate-Dienstes hinter einem Load Balancer und einer CI/CD Pipeline erforderlich sind, die verwendet. AWS CodePipeline

Um eine Service-Vorlage zu registrieren

1. Erstellen Sie eine Dienstvorlage.

   Die Dienstvorlagenressource verfolgt die Versionen der Dienstvorlagen.

```
$ 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. Erstellen Sie eine Konfiguration für die Vorlagensynchronisierung.

   AWS Proton richtet eine Synchronisierungsbeziehung zwischen Ihrem Repository und Ihrer Service-Vorlage ein. Anschließend wird die Vorlagenversion 1.0 im `DRAFT` Status erstellt.

```
$ 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. Warten Sie, bis die Version der Dienstvorlage erfolgreich registriert wurde.

   Wenn dieser Befehl mit dem Exit-Status von zurückkehrt`0`, ist die Versionsregistrierung abgeschlossen. Dies ist in Skripten nützlich, um sicherzustellen, dass Sie den Befehl im nächsten Schritt erfolgreich ausführen können.

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

4. Veröffentlichen Sie die Version der Dienstvorlage, um sie für die Diensterstellung verfügbar zu machen.

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

Schritt 3: Stellen Sie eine Umgebung bereit

In diesem Schritt instanziieren Sie als Administrator eine AWS Proton Umgebung anhand der Umgebungsvorlage.

Um eine Umgebung bereitzustellen

1. Holen Sie sich eine Beispielspezifikationsdatei für die Umgebungsvorlage, die Sie registriert haben.

   Sie können die Datei `environment-templates/fargate-env/spec/spec.yaml` aus dem Vorlagen-Beispiel-Repository herunterladen. Alternativ können Sie das gesamte Repository lokal abrufen und den create-environment Befehl aus dem `environment-templates/fargate-env` Verzeichnis ausführen.

2. Erstellen Sie eine Umgebung.

AWS Proton liest Eingabewerte aus Ihrer Umgebungsspezifikation, kombiniert sie mit Ihrer Umgebungsvorlage und stellt mithilfe Ihrer AWS Proton Servicerolle Umgebungsressourcen in Ihrem AWS Konto bereit.

```
$ 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. Warten Sie, bis die Umgebung erfolgreich bereitgestellt wurde.

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

Schritt 4: Bereitstellen eines Dienstes [Anwendungsentwickler]

In den vorherigen Schritten hat ein Administrator eine Dienstvorlage registriert und veröffentlicht und eine Umgebung bereitgestellt. Als Anwendungsentwickler können Sie jetzt einen AWS Proton Dienst erstellen und ihn in der AWS Proton Umgebung bereitstellen

Um einen Dienst bereitzustellen

1. Rufen Sie eine Beispielspezifikationsdatei für die Dienstvorlage ab, die der Administrator registriert hat.

   Sie können die Datei `service-templates/load-balanced-fargate-svc/spec/spec.yaml` aus dem Vorlagen-Beispiel-Repository herunterladen. Alternativ können Sie das gesamte Repository lokal abrufen und den create-service Befehl aus dem `service-templates/load-balanced-fargate-svc` Verzeichnis ausführen.
2. Teilen Sie das AWS Proton Sample Services-Repository Ihrem GitHub Konto oder Ihrer Organisation zu. Dieses Repository enthält den Quellcode der Anwendung, den wir in diesem Tutorial verwenden.
3. Erstellen Sie einen Service.

   AWS Proton liest Eingabewerte aus Ihrer Servicespezifikation, kombiniert sie mit Ihrer Dienstvorlage und stellt Serviceressourcen in Ihrem AWS Konto in der Umgebung bereit, die in

der Spezifikation angegeben ist. Eine AWS CodePipeline Pipeline stellt Ihren Anwendungscode aus dem Repository bereit, das Sie im Befehl angeben.

```
$ 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. Warten Sie, bis der Dienst erfolgreich bereitgestellt wurde.

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

5. Rufen Sie die Ausgaben ab und sehen Sie sich Ihre neue Website an.

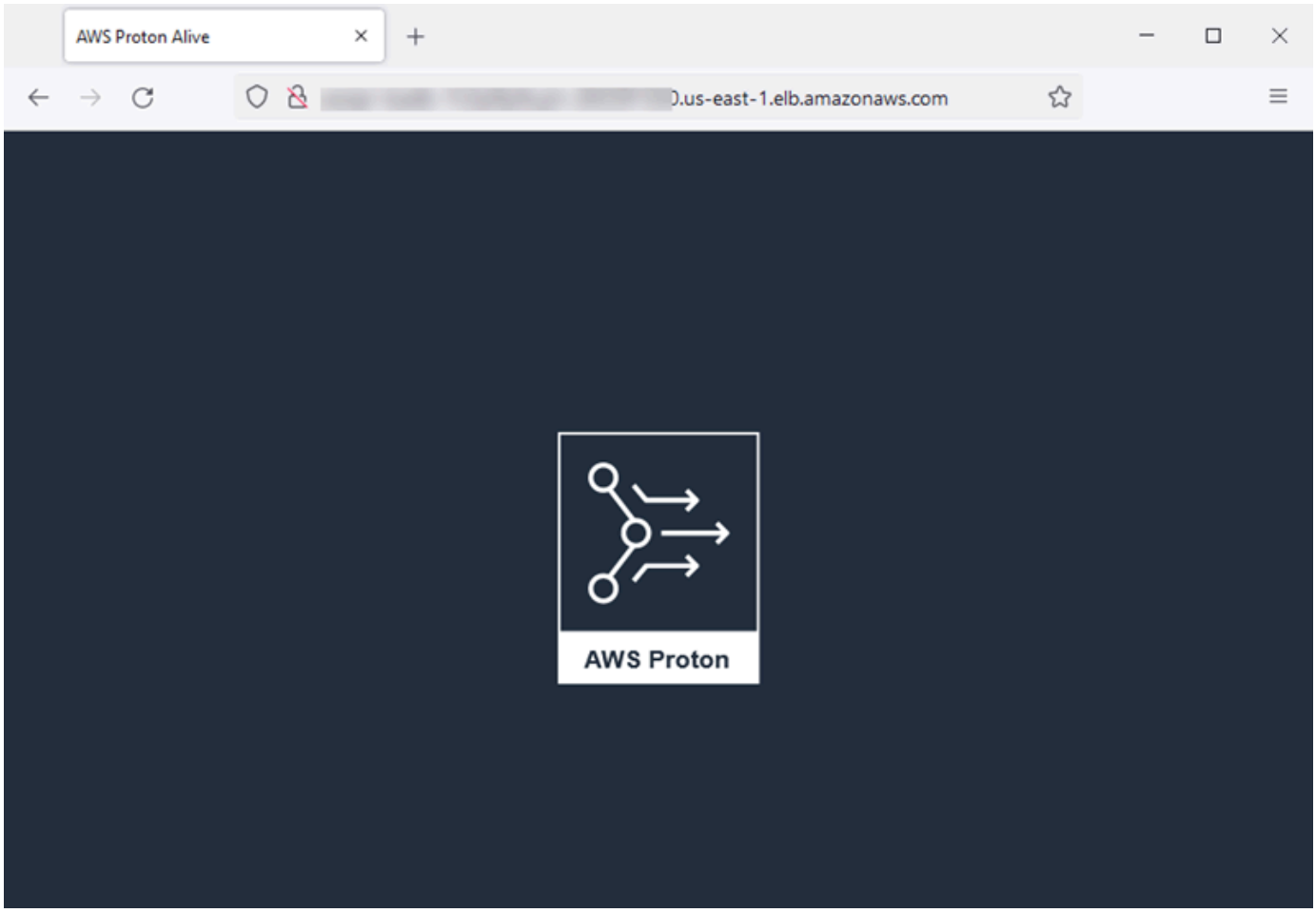
   Führen Sie den folgenden Befehl aus:

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

   Die Ausgabe des Befehls sollte der folgenden ähneln:

```
{
    "outputs": [
        {
            "key": "ServiceURL",
            "valueString": "http://your-service-endpoint.us-east-1.elb.amazonaws.com"
        }
    ]
}
```

   Der Wert der `ServiceURL` Instanzausgabe ist der Endpunkt Ihrer neuen Service-Website. Verwenden Sie Ihren Browser, um dorthin zu navigieren. Sie sollten die folgende Grafik auf einer statischen Seite sehen:



Schritt 5: Aufräumen (optional)

In diesem Schritt löschen Sie die AWS Ressourcen, die Sie im Rahmen dieses Tutorials erstellt haben, und um die mit diesen Ressourcen verbundenen Kosten zu sparen.

Um Tutorial-Ressourcen zu löschen

1. Führen Sie den folgenden Befehl aus, um den Dienst zu löschen:

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

2. Führen Sie den folgenden Befehl aus, um die Umgebung zu löschen:

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

3. Führen Sie die folgenden Befehle aus, um die Dienstvorlage zu löschen:

```
$ 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. Führen Sie die folgenden Befehle aus, um die Umgebungsvorlage zu löschen:

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

Die AWS Proton Vorlagenbibliothek

Das AWS Proton Team unterhält eine Bibliothek mit Vorlagenbeispielen auf GitHub. Die Bibliothek enthält Beispiele für Infrastructure-as-Code-Dateien (IaC) für viele gängige Umgebungs- und Anwendungsinfrastrukturszenarien.

Die Vorlagenbibliothek ist in zwei GitHub Repositorys gespeichert:

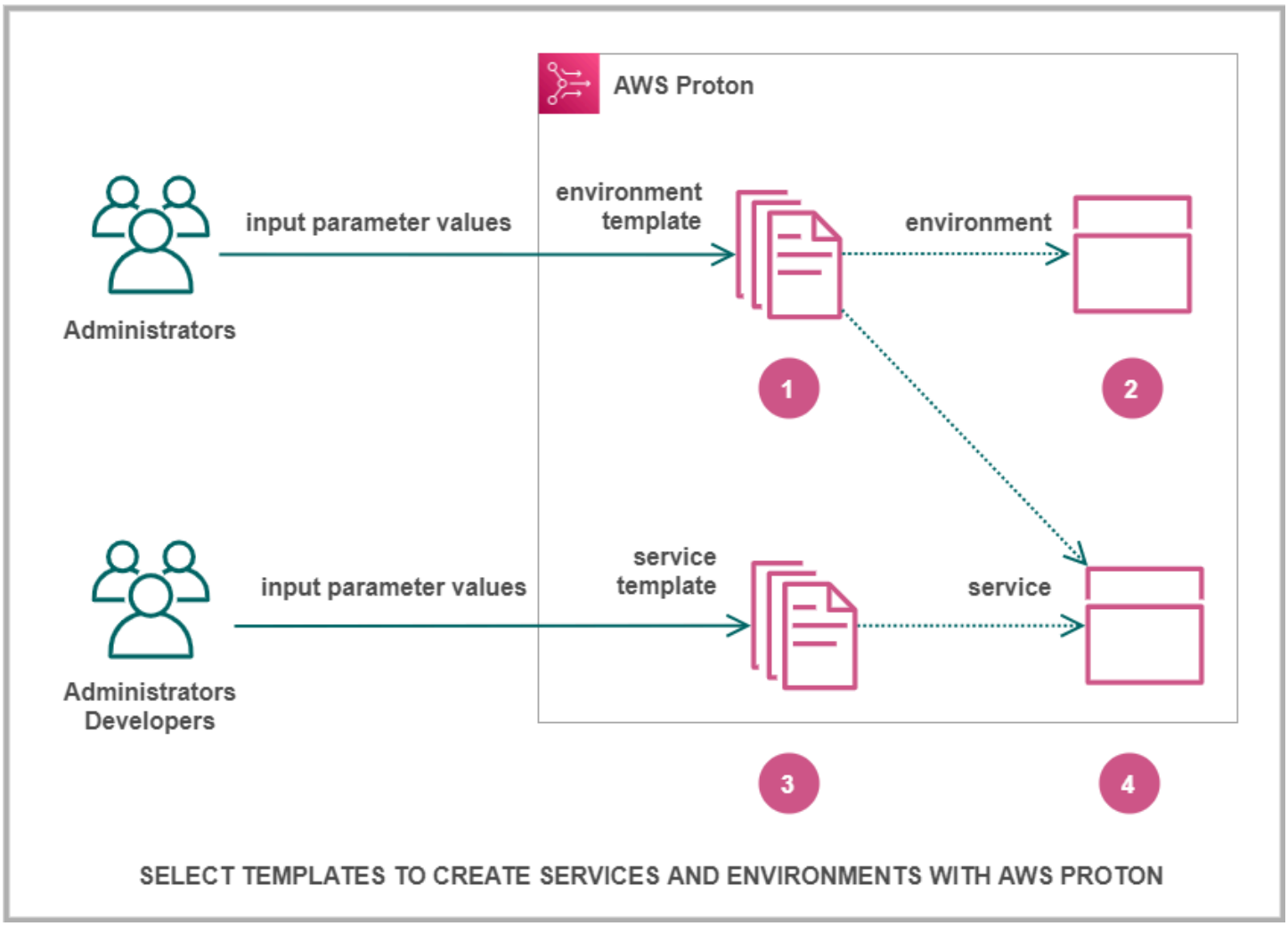
- aws-proton-cloudformation-sample-templates — Beispiele für Template-Bundles, die Jinja als AWS CloudFormationIaC-Sprache verwenden. Sie können diese Beispiele für Umgebungen verwenden. AWS-verwaltete Bereitstellung
- aws-proton-terraform-sample-templates — Beispiele für Vorlagenpakete, die Terraform als IaC-Sprache verwenden. Sie können diese Beispiele für Umgebungen verwenden. Selbstverwaltete Bereitstellung

Jedes dieser Repositorien enthält eine `README` Datei mit vollständigen Informationen über den Inhalt und die Struktur des Repositorys. Jedes Beispiel enthält Informationen über den Anwendungsfall, den die Vorlage abdeckt, die Architektur des Beispiels und die Eingabeparameter, die die Vorlage verwendet.

Sie können die Vorlagen in dieser Bibliothek direkt verwenden, indem Sie eines der Repositorys der Bibliothek in Ihr GitHub Konto einbinden. Alternativ können Sie diese Beispiele als Ausgangspunkt für die Entwicklung Ihrer Umgebungs- und Dienstvorlagen verwenden.

Wie AWS Proton funktioniert

Mit AWS Proton stellen Sie Umgebungen bereit und anschließend Dienste, die in diesen Umgebungen ausgeführt werden. Umgebungen und Dienste basieren auf Umgebungs- bzw. Dienstvorlagen, die Sie in Ihrer AWS Proton versionierten Vorlagenbibliothek auswählen.



1

Wenn Sie als Administrator eine Umgebungsvorlage mit auswählen AWS Proton, geben Sie Werte für die erforderlichen Eingabeparameter an.

2

AWS Proton verwendet die Umgebungsvorlage und die Parameterwerte, um Ihre Umgebung bereitzustellen.

3

Wenn Sie als Entwickler oder Administrator eine Dienstvorlage mit auswählen AWS Proton, geben Sie Werte für die erforderlichen Eingabeparameter an. Sie wählen auch eine Umgebung aus, in der Sie Ihre Anwendung oder Ihren Dienst bereitstellen möchten.

4

AWS Proton verwendet die Dienstvorlage und sowohl Ihren Service als auch die ausgewählten Umgebungsparameterwerte, um Ihren Service bereitzustellen.

Sie geben Werte für die Eingabeparameter an, um Ihre Vorlage für die Wiederverwendung und mehrere Anwendungsfälle, Anwendungen oder Dienste anzupassen.

Damit dies funktioniert, erstellen Sie Umgebungs- oder Dienstvorlagenpakete und laden sie in registrierte Umgebungs- bzw. Dienstvorlagen hoch.

Vorlagenpakete enthalten alles, was für die Bereitstellung von Umgebungen oder Diensten AWS Proton erforderlich ist.

Wenn Sie eine Umgebungs- oder Dienstvorlage erstellen, laden Sie ein Vorlagenpaket hoch, das die parametrisierten IaC-Dateien (Infrastructure as Code) enthält, die zur Bereitstellung von Umgebungen oder Diensten AWS Proton verwendet werden.

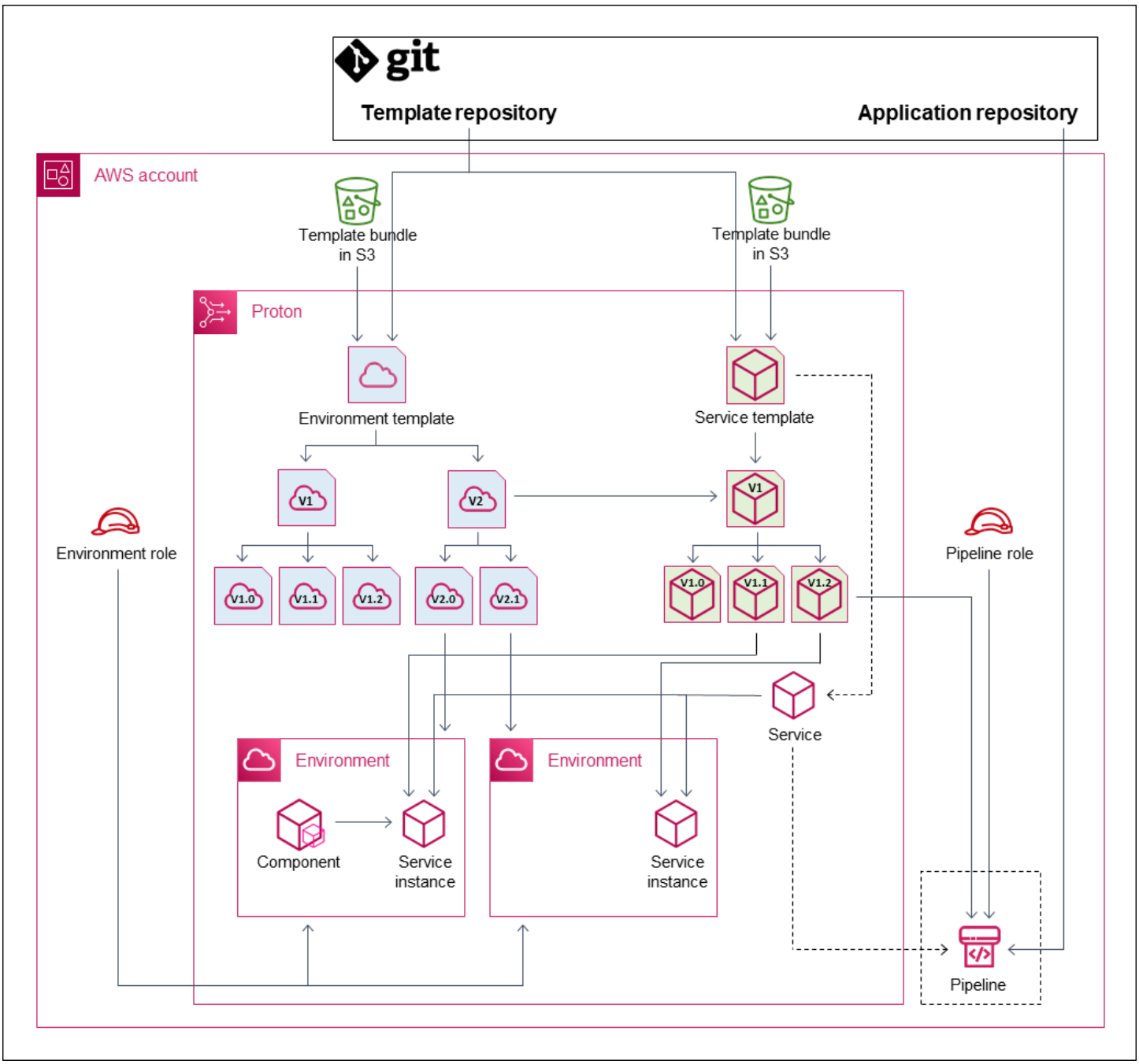
Wenn Sie eine Umgebungs- oder Dienstvorlage auswählen, um eine Umgebung oder einen Dienst zu erstellen oder zu aktualisieren, geben Sie Werte für die IaC-Dateiparameter des Vorlagenpakets an.

Themen

AWS Proton Objekte

Das folgende Diagramm zeigt die AWS Proton Hauptobjekte und ihre Beziehung zu anderen Objekten AWS und Objekten von Drittanbietern. Die Pfeile stellen die Richtung des Datenflusses dar (die umgekehrte Richtung der Abhängigkeit).

Wir folgen dem Diagramm mit kurzen Beschreibungen und Referenzlinks für diese AWS Proton Objekte.



- Umgebungsvorlage — Eine Sammlung von Versionen von Umgebungsvorlagen, die zur Erstellung von Umgebungen verwendet werden können. AWS Proton

  Weitere Informationen erhalten Sie unter Erstellung von Vorlagen und Bundles und -Vorlagen.
- Version der Umgebungsvorlage — Eine bestimmte Version einer Umgebungsvorlage. Nimmt ein Vorlagenpaket als Eingabe, entweder aus einem S3-Bucket oder aus einem Git-Repository. Das Paket spezifiziert Infrastructure as Code (IaC) und zugehörige Eingabeparameter für eine AWS Proton Umgebung.

Weitere Informationen finden Sie unter the section called “Versionen”, the section called “Veröffentlichen” und the section called “Konfigurationen für die Vorlagensynchronisierung”.

- Umgebung — Der Satz gemeinsam genutzter AWS Infrastrukturressourcen und Zugriffsrichtlinien, in denen AWS Proton Dienste bereitgestellt werden. AWS Ressourcen werden mithilfe einer Umgebungsvorlagenversion bereitgestellt, die mit bestimmten Parameterwerten aufgerufen wird. Zugriffsrichtlinien werden in einer Servicerolle bereitgestellt.

  Weitere Informationen finden Sie unter Umgebungen.

- Dienstvorlage — Eine Sammlung von Dienstvorlagenversionen, die zur Erstellung von AWS Proton Diensten verwendet werden können.

  Weitere Informationen erhalten Sie unter Erstellung von Vorlagen und Bundles und -Vorlagen.

- Version der Dienstvorlage — Eine bestimmte Version einer Dienstvorlage. Nimmt ein Vorlagenpaket als Eingabe, entweder aus einem S3-Bucket oder aus einem Git-Repository. Das Paket spezifiziert Infrastructure as Code (IaC) und zugehörige Eingabeparameter für einen AWS Proton Dienst.

  Eine Dienstvorlagenversion spezifiziert auch diese Einschränkungen für Dienstinstanzen auf der Grundlage der Version:

  - Kompatible Umgebungsvorlagen — Instanzen können nur in Umgebungen ausgeführt werden, die auf diesen kompatiblen Umgebungsvorlagen basieren.
  - Unterstützte Komponentenquellen — Die Arten von Komponenten, die Entwickler mit Instanzen verknüpfen können.

  Weitere Informationen finden Sie unter the section called “Versionen”, the section called “Veröffentlichen” und the section called “Konfigurationen für die Vorlagensynchronisierung”.

- Service — Eine Sammlung von Dienstinstanzen, die eine Anwendung mithilfe von Ressourcen ausführen, die in einer Dienstvorlage angegeben sind, und optional eine CI/CD Pipeline, die den Anwendungscode in diesen Instanzen bereitstellt.

  Im Diagramm bedeutet die gestrichelte Linie aus der Dienstvorlage, dass der Dienst die Vorlage an die Dienstinstanzen und die Pipeline weiterleitet.

  Weitere Informationen finden Sie unter Dienstleistungen.

- Dienstinstanz — Die Gruppe von AWS Infrastrukturressourcen, die eine Anwendung in einer bestimmten AWS Proton Umgebung ausführen. AWS Ressourcen werden mithilfe einer Dienstvorlagenversion bereitgestellt, die mit bestimmten Parameterwerten aufgerufen wird.

  Weitere Informationen erhalten Sie unter Dienstleistungen und the section called “Instanz aktualisieren”.
- Pipeline — Eine optionale CI/CD Pipeline, die eine Anwendung in den Instanzen eines Dienstes bereitstellt und über Zugriffsrichtlinien für die Bereitstellung dieser Pipeline verfügt. Zugriffsrichtlinien werden in einer Servicerolle bereitgestellt. Einem Dienst ist nicht immer eine AWS Proton Pipeline zugeordnet — Sie können Ihre App-Code-Bereitstellungen auch außerhalb von verwalten. AWS Proton

  Im Diagramm bedeuten die gestrichelte Linie von Service und das gestrichelte Feld rund um Pipeline, dass, wenn Sie Ihre CI/CD Bereitstellungen selbst verwalten möchten, die AWS Proton Pipeline möglicherweise nicht erstellt wird und sich Ihre eigene Pipeline möglicherweise nicht in Ihrem Konto befindet. AWS

  Weitere Informationen erhalten Sie unter Dienstleistungen und the section called “Pipeline aktualisieren”.
- Komponente — Eine vom Entwickler definierte Erweiterung einer Dienstinstanz. Gibt zusätzliche AWS Infrastrukturressourcen an, die eine bestimmte Anwendung möglicherweise zusätzlich zu den Ressourcen benötigt, die von der Umgebung und der Dienstinstanz bereitgestellt werden. Plattformteams kontrollieren die Infrastruktur, die eine Komponente bereitstellen kann, indem sie der Umgebung eine Komponentenrolle zuweisen.

  Weitere Informationen finden Sie unter Komponenten.

Wie stellt die AWS Proton Infrastruktur bereit

AWS Proton kann die Infrastruktur auf eine von mehreren Arten bereitstellen:

- AWS-verwaltete Bereitstellung — AWS Proton ruft die Provisioning-Engine in Ihrem Namen auf. Diese Methode unterstützt nur AWS CloudFormation Vorlagenpakete. Weitere Informationen finden Sie unter the section called “CloudFormation IaC-Dateien”.
- CodeBuild Bereitstellung — dient AWS CodeBuild zur Ausführung AWS Proton von Shell-Befehlen, die Sie bereitstellen. Ihre Befehle können Eingaben lesen, die die Infrastruktur bereitstellen, und sind für die AWS Proton Bereitstellung oder Deprovisionierung der Infrastruktur und die

Generierung von Ausgabewerten verantwortlich. Ein Vorlagenpaket für diese Methode enthält Ihre Befehle in einer Manifestdatei sowie alle Programme, Skripts oder anderen Dateien, die diese Befehle möglicherweise benötigen.

Als Beispiel für die Verwendung von CodeBuild Provisioning können Sie Code hinzufügen, der die AWS Cloud Development Kit (AWS CDK) zur Bereitstellung von AWS Ressourcen verwendet, und ein Manifest, das das CDK installiert und Ihren CDK-Code ausführt.

Weitere Informationen finden Sie unter the section called “CodeBuild bündeln”.

> **Note**
>
> Sie können die CodeBuild Bereitstellung mit Umgebungen und Diensten verwenden. Derzeit können Sie Komponenten auf diese Weise nicht bereitstellen.

- Selbstverwaltete Bereitstellung — AWS Proton sendet eine Pull-Anfrage (PR) an ein von Ihnen bereitgestelltes Repository, in dem Ihr eigenes Infrastrukturbereitstellungssystem den Bereitstellungsprozess durchführt. Diese Methode unterstützt nur Terraform-Vorlagenpakete. Weitere Informationen finden Sie unter the section called “Terraform-IaC-Dateien”.

AWS Proton bestimmt und legt die Bereitstellungsmethode für jede Umgebung und jeden Dienst separat fest. Wenn Sie eine Umgebung oder einen Dienst erstellen oder aktualisieren, AWS Proton untersucht das von Ihnen bereitgestellte Vorlagenpaket und bestimmt die Bereitstellungsmethode, die im Vorlagenpaket angegeben ist. Auf Umgebungsebene geben Sie die Parameter an, die die Umgebung und ihre potenziellen Dienste möglicherweise für ihre Bereitstellungsmethoden benötigen —AWS Identity and Access Management (IAM) -Rollen, eine Umgebungskontoverbindung oder ein Infrastruktur-Repository.

Entwickler, die einen Dienst bereitstellen AWS Proton , haben unabhängig von der Bereitstellungsmethode die gleiche Erfahrung. Entwickler müssen sich der Bereitstellungsmethode nicht bewusst sein und müssen nichts am Servicebereitstellungsprozess ändern. Die Dienstvorlage legt die Bereitstellungsmethode fest, und jede Umgebung, in der ein Entwickler den Dienst bereitstellt, stellt die erforderlichen Parameter für die Bereitstellung von Dienstinstanzen bereit.

Das folgende Diagramm fasst einige Hauptmerkmale der verschiedenen Bereitstellungsmethoden zusammen. Die Abschnitte, die der Tabelle folgen, enthalten Einzelheiten zu den einzelnen Methoden.

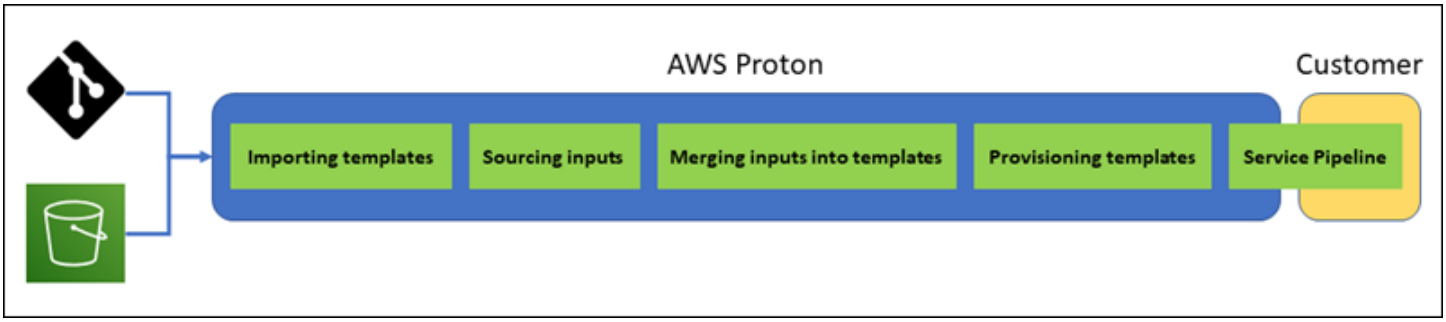
Bereitste llungsmethode	-Vorlagen	Bereitgestellt von	Status verfolgt von
AWS-verwaltet	Manifest, Schema, IaC-Datei () CloudFormation	AWS Proton (durch CloudFormation)	AWS Proton (durch CloudFormation)
CodeBuild	Manifest (mit Befehlen) , Schema, Befehlsab hängigkeiten (z. B. AWS CDK Code)	AWS Proton (durch CodeBuild)	AWS Proton (Ihre Befehle geben den Status zurück durch CodeBuild)
selbstverwaltet	Manifest-, Schema-, IaC-Dateien (Terraform)	Dein Code (durch Git-Aktionen)	Dein Code (an den er AWS per API-Aufruf weitergegeben wurde)

So funktioniert AWS-managed Provisioning

Wenn eine Umgebung oder ein Dienst AWS-verwaltetes Provisioning verwendet, wird die Infrastruktur wie folgt bereitgestellt:

1. Ein AWS Proton Kunde (ein Administrator oder ein Entwickler) erstellt die AWS Proton Ressource (eine Umgebung oder einen Dienst). Der Kunde wählt eine Vorlage für die Ressource aus und gibt die erforderlichen Parameter an. Weitere Informationen finden Sie im folgenden Abschnitt,the section called “Überlegungen zur -verwalteten Bereitstellung AWS”.
2. AWS Proton rendert eine vollständige CloudFormation Vorlage für die Bereitstellung der Ressource.
3. AWS Proton ruft CloudFormation auf, um die Bereitstellung mithilfe der gerenderten Vorlage zu starten.
4. AWS Proton überwacht kontinuierlich die CloudFormation Bereitstellung.
5. Wenn die Bereitstellung abgeschlossen ist, AWS Proton meldet es Fehler im Falle eines Fehlers und erfasst im Erfolgsfall Bereitstellungsausgaben wie die Amazon VPC-ID.

Das folgende Diagramm zeigt, dass die meisten dieser Schritte direkt AWS Proton erledigt werden.



Überlegungen zur -verwalteten Bereitstellung AWS

- Rolle für die Infrastrukturbereitstellung — Wenn eine Umgebung oder eine der darin ausgeführten Dienstinstanzen möglicherweise AWS-managed Provisioning verwendet, muss ein Administrator eine IAM-Rolle konfigurieren (entweder direkt oder als Teil einer Umgebungskontoverbindung). AWS Proton AWS Proton verwendet diese Rolle, um die Infrastruktur dieser AWS verwalteten Bereitstellungsressourcen bereitzustellen. Die Rolle sollte über die erforderlichen Berechtigungen verfügen, CloudFormation um alle Ressourcen zu erstellen, die in den Vorlagen dieser Ressourcen enthalten sind.

  Weitere Informationen erhalten Sie unter the section called “IAM-Rollen” und the section called “Beispiele für Richtlinien für Servicerollen”.
- Servicebereitstellung — Wenn ein Entwickler eine Dienstinstanz, die AWS-managed Provisioning verwendet, in der Umgebung bereitstellt, AWS Proton verwendet er die dieser Umgebung zur Verfügung gestellte Rolle, um die Infrastruktur für die Dienstinstanz bereitzustellen. Entwickler sehen diese Rolle nicht und können sie auch nicht ändern.
- Dienst mit Pipeline — Eine Dienstvorlage, die AWS-managed Provisioning verwendet, kann eine Pipeline-Definition enthalten, die im CloudFormation YAML-Schema geschrieben ist. AWS Proton erstellt die Pipeline auch durch Aufrufen. CloudFormation Die Rolle, die zum Erstellen einer Pipeline AWS Proton verwendet wird, unterscheidet sich von der Rolle für jede einzelne Umgebung. Diese Rolle wird AWS Proton separat, nur einmal auf AWS Kontoebene, bereitgestellt und dient zur Bereitstellung und Verwaltung aller AWS verwalteten Pipelines. Diese Rolle sollte über Berechtigungen zum Erstellen von Pipelines und anderen Ressourcen verfügen, die Ihre Pipelines benötigen.

  Die folgenden Verfahren zeigen, wie Sie die Pipeline-Rolle bereitstellen. AWS Proton

AWS Proton console

Um die Pipeline-Rolle bereitzustellen

1. Wählen Sie in der AWS Proton Konsole im Navigationsbereich Einstellungen > Kontoeinstellungen und dann Konfigurieren aus.
2. Verwenden Sie den Abschnitt „ AWS Pipeline-verwaltete Rolle“, um eine neue oder bestehende Pipeline-Rolle für die AWS-verwaltete Bereitstellung zu konfigurieren.

AWS Proton API

Um die Pipeline-Rolle bereitzustellen

1. Verwenden Sie die UpdateAccountSettingsAPI-Aktion.
2. Geben Sie den Amazon-Ressourcennamen (ARN) Ihrer Pipeline-Servicerolle in den `pipelineServiceRoleArn` Parameter ein.

AWS CLI

Um die Pipeline-Rolle bereitzustellen

Führen Sie den folgenden Befehl aus:

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

Wie CodeBuild funktioniert die Bereitstellung

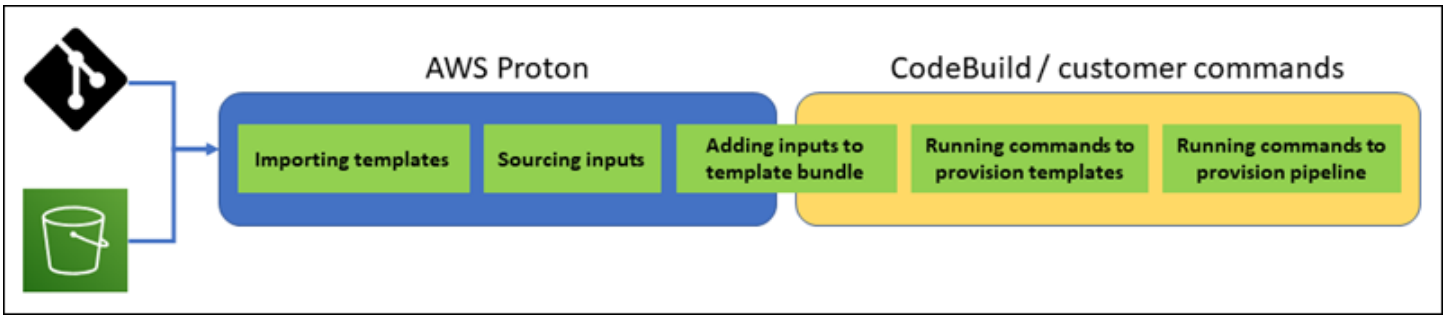
Wenn eine Umgebung oder ein Dienst CodeBuild Provisioning verwendet, wird die Infrastruktur wie folgt bereitgestellt:

1. Ein AWS Proton Kunde (ein Administrator oder ein Entwickler) erstellt die AWS Proton Ressource (eine Umgebung oder einen Dienst). Der Kunde wählt eine Vorlage für die Ressource aus und gibt die erforderlichen Parameter an. Weitere Informationen finden Sie im folgenden Abschnitt,the section called “Überlegungen zur Bereitstellung CodeBuild”.

2. AWS Proton rendert eine Eingabedatei mit Eingabeparameterwerten für die Bereitstellung der Ressource.
3. AWS Proton ruft CodeBuild auf, um einen Job zu starten. Der CodeBuild Job führt die in der Vorlage angegebenen Shell-Befehle des Kunden aus. Diese Befehle stellen die gewünschte Infrastruktur bereit und lesen optional Eingabewerte.
4. Wenn die Bereitstellung abgeschlossen ist, gibt der letzte Kundenbefehl den Bereitstellungsstatus auf zurück CodeBuild und ruft die NotifyResourceDeploymentStatusChange AWS Proton API-Aktion auf, um Ausgaben wie die Amazon VPC-ID bereitzustellen, falls vorhanden.

> **Important**
>
> Stellen Sie sicher, dass Ihre Befehle den Bereitstellungsstatus korrekt zurückgeben CodeBuild und die Ausgaben bereitstellen. Wenn dies nicht der Fall ist, AWS Proton kann der Bereitstellungsstatus nicht ordnungsgemäß nachverfolgt werden und es können keine korrekten Ausgaben für Serviceinstanzen bereitgestellt werden.

Das folgende Diagramm zeigt die Schritte, die AWS Proton ausgeführt werden, und die Schritte, die Ihre Befehle innerhalb eines CodeBuild Jobs ausführen.



Überlegungen zur Bereitstellung CodeBuild

- Rolle für die Infrastrukturbereitstellung — Wenn eine Umgebung oder eine der darin ausgeführten Dienstinstanzen die CodeBuild basierte Bereitstellung verwenden könnte, muss ein Administrator eine IAM-Rolle konfigurieren (entweder direkt oder als Teil einer AWS Proton Umgebungskontoverbindung). AWS Proton verwendet diese Rolle, um die Infrastruktur dieser CodeBuild Bereitstellungsressourcen bereitzustellen. Die Rolle sollte über die erforderlichen Berechtigungen verfügen CodeBuild , um alle Ressourcen zu erstellen, die Sie in den Vorlagen für diese Ressourcen bereitstellen.

Weitere Informationen erhalten Sie unter the section called “IAM-Rollen” und the section called “Beispiele für Richtlinien für Servicerollen”.

- Servicebereitstellung — Wenn ein Entwickler eine Dienstinstanz bereitstellt, die CodeBuild Provisioning für die Umgebung verwendet, AWS Proton verwendet er die dieser Umgebung zur Verfügung gestellte Rolle, um die Infrastruktur für die Dienstinstanz bereitzustellen. Entwickler sehen diese Rolle nicht und können sie auch nicht ändern.
- Service mit Pipeline — Eine Dienstvorlage, die CodeBuild Provisioning verwendet, kann Befehle zur Bereitstellung einer Pipeline enthalten. AWS Proton erstellt die Pipeline auch durch Aufrufen CodeBuild. Die Rolle, die zum Erstellen einer Pipeline AWS Proton verwendet wird, unterscheidet sich von der Rolle für jede einzelne Umgebung. Diese Rolle wird AWS Proton separat, nur einmal auf AWS Kontoebene, bereitgestellt und dient zur Bereitstellung und Verwaltung aller CodeBuild basierten Pipelines. Diese Rolle sollte über Berechtigungen zum Erstellen von Pipelines und anderen Ressourcen verfügen, die Ihre Pipelines benötigen.

  Die folgenden Verfahren zeigen, wie Sie die Pipeline-Rolle bereitstellen. AWS Proton

  AWS Proton console

  Um die Pipeline-Rolle bereitzustellen

  1. Wählen Sie in der AWS Proton Konsole im Navigationsbereich Einstellungen > Kontoeinstellungen und dann Konfigurieren aus.
  2. Verwenden Sie den Abschnitt Codebuild-Pipeline-Bereitstellungsrolle, um eine neue oder bestehende Pipeline-Rolle für CodeBuild die Bereitstellung zu konfigurieren.

  AWS Proton API

  Um die Pipeline-Rolle bereitzustellen

  1. Verwenden Sie die UpdateAccountSettingsAPI-Aktion.
  2. Geben Sie den Amazon-Ressourcennamen (ARN) Ihrer Pipeline-Servicerolle in den `pipelineCodebuildRoleArn` Parameter ein.

  AWS CLI

  Um die Pipeline-Rolle bereitzustellen

Führen Sie den folgenden Befehl aus:

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

Wie funktioniert die selbstverwaltete Bereitstellung

Wenn eine Umgebung für die Verwendung von selbstverwalteter Bereitstellung konfiguriert ist, wird die Infrastruktur wie folgt bereitgestellt:

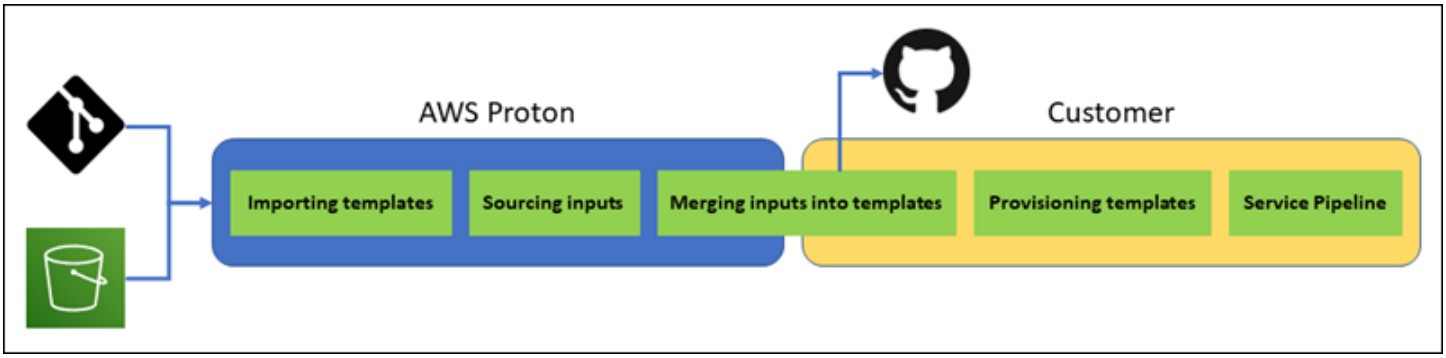
1. Ein AWS Proton Kunde (ein Administrator oder ein Entwickler) erstellt die AWS Proton Ressource (eine Umgebung oder einen Dienst). Der Kunde wählt eine Vorlage für die Ressource aus und gibt die erforderlichen Parameter an. Für eine Umgebung stellt der Kunde auch ein verknüpftes Infrastruktur-Repository zur Verfügung. Weitere Informationen finden Sie im folgenden Abschnitt,the section called “Überlegungen zur selbstverwalteten Bereitstellung”.
2. AWS Proton rendert eine vollständige Terraform-Vorlage. Es besteht aus einer oder mehreren Terraform-Dateien, möglicherweise in mehreren Ordnern, und einer Variablendatei. `.tfvars` AWS Proton schreibt Parameterwerte, die beim Aufruf zur Ressourcenerstellung bereitgestellt wurden, in diese Variablendatei.
3. AWS Proton sendet eine PR mit der gerenderten Terraform-Vorlage an das Infrastruktur-Repository.
4. Wenn der Kunde (Administrator oder Entwickler) den PR zusammenführt, veranlasst die Automatisierung des Kunden die Provisioning-Engine, die Bereitstellung der Infrastruktur mithilfe der zusammengeführten Vorlage zu starten.

   > **Note**
   >
   > Wenn der Kunde (Administrator oder Entwickler) den PR schließt, AWS Proton erkennt er den PR als geschlossen an und markiert die Bereitstellung als storniert.

5. Wenn die Bereitstellung abgeschlossen ist, ruft die Automatisierung des Kunden die NotifyResourceDeploymentStatusChange AWS Proton API-Aktion auf, um den Abschluss anzuzeigen, den Status (erfolgreich oder fehlgeschlagen) anzugeben und Ausgaben wie die Amazon VPC-ID bereitzustellen, falls vorhanden.

> **Important**
>
> Stellen Sie sicher, dass Ihr Automatisierungscode AWS Proton mit dem Bereitstellungsstatus und den Ergebnissen zurückruft. Wenn dies nicht der Fall ist, AWS Proton können Sie die Bereitstellung länger als geplant als ausstehend betrachten und weiterhin den Status In Bearbeitung anzeigen.

Das folgende Diagramm zeigt die Schritte, die dabei ausgeführt werden, und die Schritte, die Ihr eigenes Bereitstellungssystem AWS Proton ausführt.



Überlegungen zur selbstverwalteten Bereitstellung

- Infrastruktur-Repository — Wenn ein Administrator eine Umgebung für die selbstverwaltete Bereitstellung konfiguriert, muss er ein verknüpftes Infrastruktur-Repository bereitstellen. AWS Proton übermittelt PRs an dieses Repository, um die Infrastruktur der Umgebung und alle darin bereitgestellten Dienstinstanzen bereitzustellen. Die kundeneigene Automatisierungsaktion im Repository sollte eine IAM-Rolle annehmen, die berechtigt ist, alle Ressourcen zu erstellen, die in Ihrer Umgebung und Ihren Dienstvorlagen enthalten sind, sowie über eine Identität, die das Zielkonto widerspiegelt. AWS Ein Beispiel für eine GitHub Aktion, die eine Rolle annimmt, finden Sie unter Eine Rolle annehmen in der Dokumentation zur GitHub Aktion „ AWS Anmeldeinformationen konfigurieren“.
- Berechtigungen — Ihr Bereitstellungscode muss sich bei Bedarf mit einem Konto authentifizieren (z. B. bei einem AWS Konto authentifizieren) und die Autorisierung für die Ressourcenbereitstellung bereitstellen (z. B. eine Rolle angeben).
- Servicebereitstellung — Wenn ein Entwickler eine Dienstinstanz bereitstellt, die selbstverwaltete Bereitstellung in der Umgebung verwendet, AWS Proton sendet er eine PR an das Repository, das

der Umgebung zugeordnet ist, um die Infrastruktur für die Dienstinstanz bereitzustellen. Entwickler sehen das Repository nicht und können es nicht ändern.

> **Note**
>
> Entwickler, die Dienste erstellen, verwenden unabhängig von der Bereitstellungsmethode denselben Prozess, und der Unterschied wird von ihnen abstrahiert. Bei der selbstverwalteten Bereitstellung reagieren Entwickler jedoch möglicherweise langsamer, da sie warten müssen, bis jemand (möglicherweise nicht sie selbst) die PR im Infrastruktur-Repository zusammenführt, bevor die Bereitstellung beginnen kann.

- Service mit Pipeline — Eine Dienstvorlage für eine Umgebung mit selbstverwalteter Bereitstellung kann eine Pipeline-Definition (z. B. eine AWS CodePipeline Pipeline) enthalten, die in Terraform HCL geschrieben ist. Um die Bereitstellung dieser Pipelines AWS Proton zu ermöglichen, stellt ein Administrator ein verknüpftes Pipeline-Repository für bereit. AWS Proton Bei der Bereitstellung einer Pipeline sollte die kundeneigene Automatisierungsaktion im Repository eine IAM-Rolle mit Berechtigungen zur Bereitstellung der Pipeline und eine Identität annehmen, die das Zielkonto widerspiegelt. AWS Das Pipeline-Repository und die Rolle unterscheiden sich von denen, die für jede einzelne Umgebung verwendet werden. Das verknüpfte Repository wird AWS Proton separat bereitgestellt, nur einmal auf AWS Kontoebene, und es wird zur Bereitstellung und Verwaltung aller Pipelines verwendet. Die Rolle sollte über Berechtigungen zum Erstellen von Pipelines und anderen Ressourcen verfügen, die Ihre Pipelines benötigen.

  Die folgenden Verfahren zeigen, wie Sie das Pipeline-Repository und die Rolle für die Pipeline bereitstellen. AWS Proton

  AWS Proton console

  Um die Pipeline-Rolle bereitzustellen

  1. Wählen Sie in der AWS Proton Konsole im Navigationsbereich Einstellungen > Kontoeinstellungen und dann Konfigurieren aus.
  2. Verwenden Sie den Abschnitt CI/CD-Pipeline-Repository, um einen neuen oder vorhandenen Repository-Link zu konfigurieren.

AWS Proton API

Um die Pipeline-Rolle bereitzustellen

1. Verwenden Sie die UpdateAccountSettingsAPI-Aktion.
2. Geben Sie im `pipelineProvisioningRepository` Parameter den Anbieter, den Namen und den Zweig Ihres Pipeline-Repositorys an.

AWS CLI

Um die Pipeline-Rolle bereitzustellen

Führen Sie den folgenden Befehl aus:

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

- Löschen von selbst verwalteten bereitgestellten Ressourcen — Terraform-Module können zusätzlich zu den Ressourcendefinitionen Konfigurationselemente enthalten, die für den Terraform-Betrieb erforderlich sind. Daher AWS Proton können nicht alle Terraform-Dateien für eine Umgebung oder Dienstinstanz gelöscht werden. AWS Proton Markiert stattdessen die Dateien zum Löschen und aktualisiert eine Markierung in den PR-Metadaten. Ihre Automatisierung kann dieses Flag lesen und damit einen Terraform-Zerstörungsbefehl auslösen.

AWS Proton Terminologie

Vorlage für die Umgebung

Definiert eine gemeinsam genutzte Infrastruktur, z. B. eine VPC oder einen Cluster, die von mehreren Anwendungen oder Ressourcen verwendet wird.

Paket mit Umgebungsvorlagen

Eine Sammlung von Dateien, die Sie hochladen, um eine Umgebungsvorlage zu erstellen und zu registrieren AWS Proton. Ein Paket mit Umgebungsvorlagen enthält Folgendes:

1. Eine Schemadatei, die Infrastruktur als Codeeingabeparameter definiert.

2. Eine Infrastructure-as-Code-Datei (IaC), die eine gemeinsam genutzte Infrastruktur wie eine VPC oder einen Cluster definiert, die von mehreren Anwendungen oder Ressourcen verwendet wird.
3. Eine Manifestdatei, die die IaC-Datei auflistet.

Umgebung

Bereitgestellte gemeinsam genutzte Infrastruktur, z. B. eine VPC oder ein Cluster, die von mehreren Anwendungen oder Ressourcen verwendet wird.

Dienstvorlage

Definiert die Art der Infrastruktur, die für die Bereitstellung und Wartung einer Anwendung oder eines Microservices in einer Umgebung erforderlich ist.

Paket mit Dienstvorlagen

Eine Sammlung von Dateien, die Sie hochladen, um eine Dienstvorlage zu erstellen und zu registrieren AWS Proton. Ein Servicevorlagenpaket enthält Folgendes:

1. Eine Schemadatei, die Eingabeparameter für Infrastruktur als Code (IaC) definiert.
2. Eine IaC-Datei, die die Infrastruktur definiert, die für die Bereitstellung und Wartung einer Anwendung oder eines Microservices in einer Umgebung erforderlich ist.
3. Eine Manifestdatei, die die IaC-Datei auflistet.
4. Optional
   a. Eine IaC-Datei, die die Service-Pipeline-Infrastruktur definiert.
   b. Eine Manifestdatei, die die IaC-Datei auflistet.

Service

Bereitgestellte Infrastruktur, die für die Bereitstellung und Wartung einer Anwendung oder eines Microservices in einer Umgebung benötigt wird.

Service-Instance

Bereitgestellte Infrastruktur, die eine Anwendung oder einen Microservice in einer Umgebung unterstützt.

Servicepipeline

Bereitgestellte Infrastruktur, die eine Pipeline unterstützt.

Version der Vorlage

Eine Haupt- oder Nebenversion einer Vorlage. Weitere Informationen finden Sie unter Versionierte Vorlagen.

Eingabeparameter

In einer Schemadatei definiert und in einer IaC-Datei (Infrastructure as Code) verwendet, sodass die IaC-Datei wiederholbar und für eine Vielzahl von Anwendungsfällen verwendet werden kann.

Schemadatei

Definiert Infrastruktur als Eingabeparameter für die Codedatei.

Spezifikationsdatei

Gibt Werte für die Infrastruktur als Eingabeparameter für die Codedatei an, wie sie in einer Schemadatei definiert sind.

Manifestdatei

Listet eine Infrastruktur als Codedatei auf.

Vorlagen erstellen und Bundles erstellen für AWS Proton

AWS Proton stellt Ihnen Ressourcen auf der Grundlage von IaC-Dateien (Infrastructure as Code) bereit. Sie beschreiben die Infrastruktur in wiederverwendbaren IaC-Dateien. Um die Dateien für verschiedene Umgebungen und Anwendungen wiederverwendbar zu machen, erstellen Sie sie als Vorlagen, definieren Eingabeparameter und verwenden diese Parameter in IaC-Definitionen. Wenn Sie später eine Bereitstellungsressource (Umgebung, Dienstinstanz oder Komponente) erstellen, AWS Proton verwendet sie eine Rendering-Engine, die Eingabewerte mit einer Vorlage kombiniert, um eine IaC-Datei zu erstellen, die bereit zur Bereitstellung ist.

Administratoren erstellen die meisten Vorlagen als Vorlagenpakete, laden sie dann hoch und registrieren sie dort. AWS Proton Im Rest dieser Seite werden diese AWS Proton Vorlagenpakete beschrieben. Direkt definierte Komponenten sind eine Ausnahme — Entwickler erstellen sie und stellen IaC-Vorlagendateien direkt zur Verfügung. Weitere Informationen zu Komponenten finden Sie unter. Komponenten

Themen

Vorlagenpakete

Als Administrator erstellen und registrieren Sie Vorlagen mit AWS Proton. Sie verwenden diese Vorlagen, um Umgebungen und Dienste zu erstellen. Wenn Sie einen Dienst erstellen, AWS Proton werden Dienstinstanzen in ausgewählten Umgebungen bereitgestellt und bereitgestellt. Weitere Informationen finden Sie unter AWS Proton für Plattformteams.

Um eine Vorlage zu erstellen und zu registrieren AWS Proton, laden Sie ein Vorlagenpaket hoch, das die IaC-Dateien (Infrastructure as Code) enthält, die bereitgestellt AWS Proton werden müssen, sowie die Umgebung oder der Service.

Ein Vorlagenpaket enthält Folgendes:

- Eine Infrastructure-as-Code-Datei (IaC) mit einer Manifest-YAML-Datei, die die IaC-Datei auflistet.
- Eine Schema-YAML-Datei für die Eingabeparameterdefinitionen Ihrer IaC-Datei.

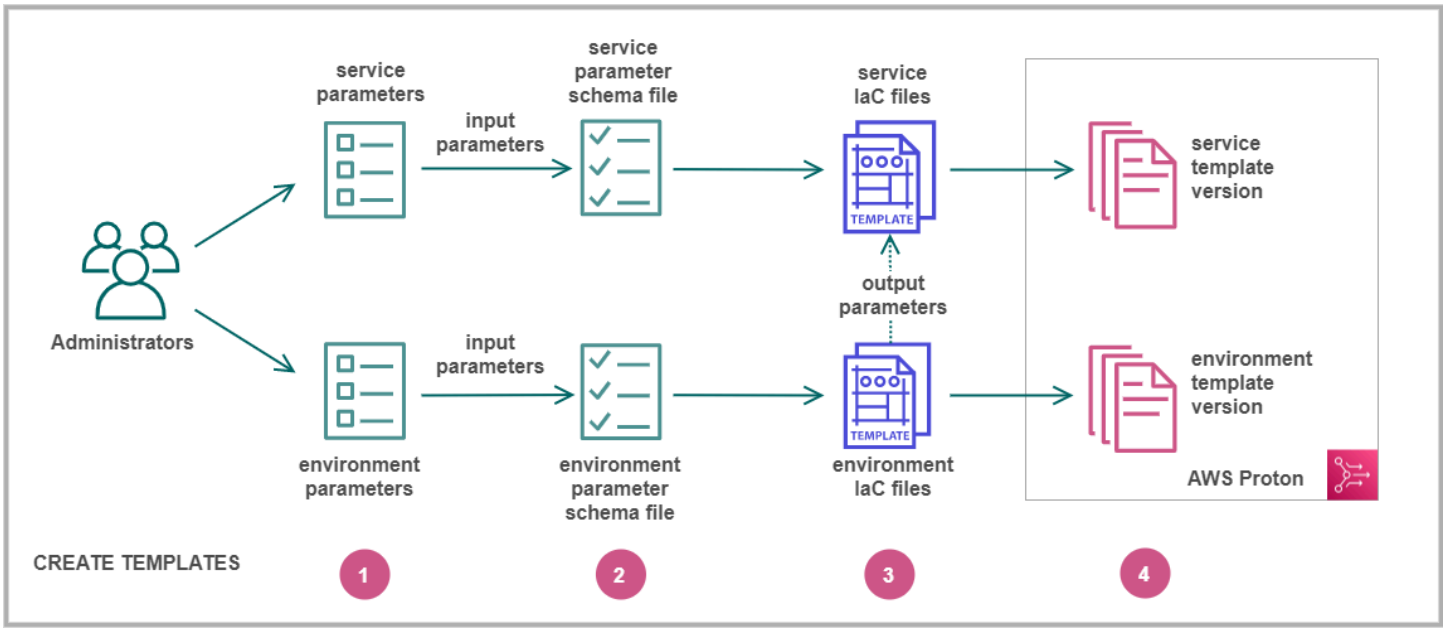
Ein Paket mit CloudFormation Umgebungsvorlagen enthält eine IaC-Datei.

Ein CloudFormation Dienstvorlagenpaket enthält eine IaC-Datei für Service-Instanzdefinitionen und eine weitere optionale IaC-Datei für eine Pipeline-Definition.

Terraform-Umgebungs- und Dienstvorlagenpakete können jeweils mehrere IaC-Dateien enthalten.

AWS Proton erfordert eine Eingabeparameter-Schemadatei. Wenn Sie Ihre IaC-Dateien erstellen, verwenden Sie die Jinja-Syntax, um auf Ihre Eingabeparameter zu verweisen. AWS CloudFormation AWS Proton stellt Parameter-Namespaces bereit, mit denen Sie auf Parameter in Ihren IaC-Dateien verweisen können.

Das folgende Diagramm zeigt ein Beispiel für Schritte, für die Sie eine Vorlage erstellen können. AWS Proton



1

Identifi

Sie die Eingabeparameter.

2

Erstelle

Sie eine Schemadatei, um Ihre Eingabeparameter zu definieren.

3 Erstelle

Sie IAc-Dateien, die auf Ihre Eingabeparameter verweisen. Sie können die Ausgaben der Umgebungs-IaC-Dateien als Eingaben für Ihre Service-IaC-Dateien referenzieren.

4 Registr

Sie eine Vorlagenversion bei AWS Proton und laden Sie Ihr Vorlagenpaket hoch.

AWS Proton Parameter

Sie können Parameter in Ihrer Infrastruktur als Codedateien (IaC) definieren und verwenden, um sie flexibel und wiederverwendbar zu machen. Sie lesen einen Parameterwert in Ihren IaC-Dateien, indem Sie auf den Namen des Parameters im Parameter-Namespace verweisen. AWS Proton AWS Proton fügt Parameterwerte in die gerenderten IaC-Dateien ein, die es bei der Ressourcenbereitstellung generiert. Verwendet Jinja, um AWS CloudFormation IaC-Parameter zu verarbeiten. AWS Proton AWS Proton Generiert zur Verarbeitung von Terraform-IaC-Parametern eine Terraform-Parameterwertdatei und stützt sich dabei auf die in HCL integrierte Parametrisierungsfunktion.

Mit AWS Proton generiert eine CodeBuild Bereitstellung Eingabedatei, die Ihr Code importieren kann. Die Datei ist eine JSON- oder HCL-Datei, abhängig von einer Eigenschaft im Manifest Ihrer Vorlage. Weitere Informationen finden Sie unter the section called “CodeBuild Bereitstellungsparameter”.

Sie können auf Parameter in Ihren Umgebungs-, Service- und Komponenten-IaC-Dateien oder im Bereitstellungscode mit den folgenden Anforderungen verweisen:

- Die Länge der einzelnen Parameternamen darf 100 Zeichen nicht überschreiten.
- Die Länge des Parameter-Namespaces und des Ressourcennamens zusammen überschreitet nicht die Zeichenbeschränkung für den Ressourcennamen.

AWS Proton Die Bereitstellung schlägt fehl, wenn diese Kontingente überschritten werden.

Parametertypen

Die folgenden Parametertypen stehen Ihnen als Referenz in AWS Proton IaC-Dateien zur Verfügung:

Eingabeparameter

Umgebungen und Dienstinstanzen können Eingabeparameter verwenden, die Sie in einer Schemadatei definieren, die Sie der Umgebung oder Dienstvorlage zuordnen. Sie können auf die Eingabeparameter einer Ressource in der IaC-Datei der Ressource verweisen. Komponenten-IaC-Dateien können sich auf Eingabeparameter der Dienstinstanz beziehen, an die die Komponente angehängt ist.

AWS Proton überprüft die Namen der Eingabeparameter anhand Ihrer Schemadatei und ordnet sie den Parametern zu, auf die in Ihren IaC-Dateien verwiesen wird, um die Eingabewerte einzufügen, die Sie bei der Ressourcenbereitstellung in eine Spezifikationsdatei eingeben.

Ausgabeparameter

Sie können Ausgaben in jeder Ihrer IaC-Dateien definieren. Eine Ausgabe kann beispielsweise ein Name, eine ID oder ein ARN einer der Ressourcen sein, die die Vorlage bereitstellt, oder sie kann eine Möglichkeit sein, eine der Eingaben der Vorlage zu übergeben. Sie können in IaC-Dateien anderer Ressourcen auf diese Ausgaben verweisen.

Definieren Sie in CloudFormation IaC-Dateien die Ausgabeparameter im `Outputs:` Block. Definieren Sie in einer Terraform-IaC-Datei jeden Ausgabeparameter mit einer Anweisung. `output`

Ressourcenparameter

AWS Proton erstellt automatisch AWS Proton Ressourcenparameter. Diese Parameter machen die Eigenschaften des AWS Proton Ressourcenobjekts verfügbar. Ein Beispiel für einen Ressourcenparameter ist`environment.name`.

Verwenden von AWS Proton Parametern in Ihren IaC-Dateien

Um einen Parameterwert in einer IaC-Datei zu lesen, verweisen Sie auf den Namen des Parameters im AWS Proton Parameter-Namespace. Für AWS CloudFormation IaC-Dateien verwenden Sie die Jinja-Syntax und umgeben den Parameter mit Paaren aus geschweiften Klammern und Anführungszeichen.

Die folgende Tabelle zeigt die Referenzsyntax für jede unterstützte Vorlagensprache mit einem Beispiel.

Sprache der Vorlage	Syntax	Beispiel: Umgebungseingabe mit dem Namen „VPC“
CloudFormation	`"{{ parameter-name }}"`	`"{{ environment.inputs.VPC }}"`
Terraform	`var.parameter-name`	`var.environment.inputs.VPC` Generierte Terraform-Variablendefinitionen

> **Note**
>
> Wenn Sie CloudFormation dynamische Parameter in Ihrer IaC-Datei verwenden, müssen Sie sie maskieren, um Fehlinterpretationsfehler durch Jinja zu vermeiden. Weitere Informationen finden Sie unter Problembehebung AWS Proton.

In der folgenden Tabelle sind die Namespace-Namen für alle Ressourcenparameter aufgeführt. AWS Proton Jeder Vorlagendateityp kann eine andere Teilmenge des Parameter-Namespaces verwenden.

Vorlagendatei	Parametertyp	Parametername	Description
Umgebung	Ressourc	`environment.` **`name`**	Environment name
	input	`environment.inputs.` *`input-name`*	Schemadefinierte Umgebungseingaben
Service	Ressourc	`environment.` **`name`** `environment.` **`account_id`**	Name und ID der Umgebung AWS-Konto
	output	`environment.outputs.` *`output-name`*	Umgebungs-IaC-Dateiausgaben
	Ressourc	`service.`**`branch_name`** `service.`**`name`**	Dienstname und Code-Repository

Vorlagendatei	Parametertyp	Parametername	Description
		`service.`**`repository_connection_arn`** `service.`**`repository_id`**	
	Ressourc	`service_instance.` **`name`**	Name der Dienstinstanz
	input	`service_instance.inputs.` *`input-name`*	Schemadifizierte Eingaben für die Dienstinstanz
	Ressourc	`service_instance.components.default.` **`name`**	Name der angehängten Standardkomponente
	output	`service_instance.components.default.outputs.` *`output-name`*	IaC-Dateiausgaben für angehängte Standardkomponenten
Pipeline	Ressourc	`service_instance.environment.` **`name`** `service_instance.environment.` **`account_id`**	Name und AWS-Konto ID der Dienstinstanzumgebung
	output	`service_instance.environment.outputs.` *`output-name`*	IaC-Dateiausgaben der Service-Instanzumgebung
	input	`pipeline.inputs.` *`input-name`*	Schemadefinierte Pipeline-Eingaben

Vorlagendatei	Parametertyp	Parametername	Description
	Ressourc	`service.`**`branch_name`** `service.`**`name`** `service.`**`repository_connection_arn`** `service.`**`repository_id`**	Dienstname und Code-Repository
	input	`service_instance.inputs.` *`input-name`*	Schemadifizierte Eingaben für Dienstinstanzen
	collection	`{% for service_instance in` **`service_instances`** `%}...{% endfor %}`	Eine Sammlung von Dienstinstanzen, die Sie durchgehen können
Komponente	Ressourc	`environment.` **`name`** `environment.` **`account_id`**	Umgebungsname und AWS-Konto Konto-ID
	output	`environment.outputs.` *`output-name`*	Umgebungs-IaC-Dateiausgaben
	Ressourc	`service.`**`branch_name`** `service.`**`name`** `service.`**`repository_connection_arn`** `service.`**`repository_id`**	Dienstname und Code-Repository (angehängte Komponenten)
	Ressourc	`service_instance.` **`name`**	Name der Dienstinstanz (angehängte Komponenten)

Vorlagendatei	Parametertyp	Parametername	Description
	input	`service_instance.inputs.` *`input-name`*	Schemadefinierte Eingaben für Dienstinstanzen (angehängte Komponenten)
	Ressourc	`component.` **`name`**	Name der Komponente

Weitere Informationen und Beispiele finden Sie in den Unterthemen zu Parametern in IaC-Vorlagendateien für verschiedene Ressourcentypen und Vorlagensprachen.

Themen

- Details und Beispiele zu den Parametern der CloudFormation IaC-Datei in der Umgebung
- Details und Beispiele zu den Parametern der CloudFormation Service-IaC-Datei
- Parameterdetails und Beispiele für die CloudFormation IaC-Komponentendatei
- Parameterfilter für CloudFormation IaC-Dateien
- CodeBuild Details und Beispiele für Bereitstellungsparameter
- Details und Beispiele für die Terraform-Infrastruktur-as-Code-Datei (IaC)

Details und Beispiele zu den Parametern der CloudFormation IaC-Datei in der Umgebung

Sie können Parameter in Ihrer Umgebungsinfrastruktur als Codedateien (IaC) definieren und referenzieren. Eine ausführliche Beschreibung der AWS Proton Parameter, Parametertypen, des Parameter-Namespace und der Verwendung von Parametern in Ihren IaC-Dateien finden Sie unter. the section called “Parameters”

Definieren Sie Umgebungsparameter

Sie können sowohl Eingabe- als auch Ausgabeparameter für IaC-Umgebungsdateien definieren.

- Eingabeparameter — Definieren Sie Eingabeparameter für die Umgebung in Ihrer Schemadatei.

Die folgende Liste enthält Beispiele für Umgebungseingabeparameter für typische Anwendungsfälle.

- VPC CIDR-Werte
- Load Balancer-Einstellungen
- Datenbank-Einstellungen
- Ein Timeout bei der Integritätsprüfung

Als Administrator können Sie Werte für Eingabeparameter angeben, wenn Sie eine Umgebung erstellen:

- Verwenden Sie die Konsole, um ein schemabasiertes Formular auszufüllen, das AWS Proton Folgendes bietet:
- Verwenden Sie die CLI, um eine Spezifikation bereitzustellen, die die Werte enthält.

- Ausgabeparameter — Definieren Sie die Umgebungsausgaben in den IaC-Dateien Ihrer Umgebung. Sie können dann in IaC-Dateien anderer Ressourcen auf diese Ausgaben verweisen.

Lesen Sie Parameterwerte in IaC-Umgebungsdateien

Sie können Parameter, die sich auf die Umgebung beziehen, in IaC-Umgebungsdateien lesen. Sie lesen einen Parameterwert, indem Sie im Parameter-Namespace auf den Namen des AWS Proton Parameters verweisen.

- Eingabeparameter — Lesen Sie einen Umgebungseingabewert, indem Sie ihn referenzieren. `environment.inputs.`*`input-name`*
- Ressourcenparameter — Lesen Sie AWS Proton Ressourcenparameter, indem Sie auf Namen wie verweisen. `environment.name`

> **Note**
>
> Für IaC-Umgebungsdateien sind keine Ausgabeparameter anderer Ressourcen verfügbar.

IaC-Beispieldateien für Umgebungen und Dienste mit Parametern

Das folgende Beispiel zeigt die Parameterdefinition und Referenz in einer IaC-Umgebungsdatei. Das Beispiel zeigt dann, wie in der IaC-Umgebungsdatei definierte Umgebungsausgabeparameter in einer Service-IaC-Datei referenziert werden können.

Example IaC-Datei für die Umgebung CloudFormation

Beachten Sie in diesem Beispiel Folgendes:

- Der `environment.inputs.` Namespace bezieht sich auf Eingabeparameter der Umgebung.
- Der Amazon EC2 Systems Manager (SSM) -Parameter `StoreInputValue` verkettet die Umgebungseingaben.
- Die `MyEnvParameterValue` Ausgabe stellt dieselbe Verkettung von Eingabeparametern wie ein Ausgabeparameter bereit. Drei zusätzliche Ausgabeparameter machen die Eingabeparameter ebenfalls einzeln verfügbar.
- Sechs zusätzliche Ausgabeparameter machen Ressourcen verfügbar, die die Umwelt bereitstellt.

```
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 CloudFormation IaC-Datei für den Dienst

Der `environment.outputs.` Namespace bezieht sich auf Umgebungsausgaben aus einer IaC-Umgebungsdatei. Beispielsweise `environment.outputs.ClusterName` liest der Name den Wert des `ClusterName` Umgebungsausgabeparameters.

```
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'
[...]
```

Details und Beispiele zu den Parametern der CloudFormation Service-IaC-Datei

Sie können Parameter in Ihrer Service- und Pipeline-Infrastruktur als Codedateien (IaC) definieren und referenzieren. Eine ausführliche Beschreibung der AWS Proton Parameter, Parametertypen, des Parameter-Namespace und der Verwendung von Parametern in Ihren IaC-Dateien finden Sie unter. the section called “Parameters”

Definieren Sie Serviceparameter

Sie können sowohl Eingabe- als auch Ausgabeparameter für Service-IaC-Dateien definieren.

- Eingabeparameter — Definieren Sie Eingabeparameter für die Dienstinstanz in Ihrer Schemadatei.

  Die folgende Liste enthält Beispiele für Service-Eingabeparameter für typische Anwendungsfälle.

  - Port
  - Aufgabengröße
  - Image
  - Gewünschte Anzahl
  - Docker-Datei
  - Befehl zum Komponententest

  Sie geben Werte für Eingabeparameter an, wenn Sie einen Dienst erstellen:

  - Verwenden Sie die Konsole, um ein schemabasiertes Formular auszufüllen, das AWS Proton Folgendes bietet:
  - Verwenden Sie die CLI, um eine Spezifikation bereitzustellen, die die Werte enthält.
- Ausgabeparameter — Definieren Sie die Ausgaben der Serviceinstanz in Ihren Service-IaC-Dateien. Sie können dann in IaC-Dateien anderer Ressourcen auf diese Ausgaben verweisen.

Lesen Sie Parameterwerte in Service-IaC-Dateien

Sie können Parameter lesen, die sich auf den Dienst und andere Ressourcen beziehen, in Service-IaC-Dateien. Sie lesen einen Parameterwert, indem Sie im Parameter-Namespace auf den Namen des AWS Proton Parameters verweisen.

- Eingabeparameter — Lesen Sie den Eingabewert einer Dienstinstanz, indem Sie auf ihn verweisen. `service_instance.inputs.`*`input-name`*
- Ressourcenparameter — Lesen Sie AWS Proton Ressourcenparameter, indem Sie auf Namen wie `service.name`service_instance.name, und verweisen. `environment.name`
- Ausgabeparameter — Lesen Sie Ausgaben anderer Ressourcen, indem Sie auf oder verweisen`environment.outputs.`*`output-name`*. `service_instance.components.default.outputs.`*`output-name`*

Beispiel für eine Service-IaC-Datei mit Parametern

Das folgende Beispiel ist ein Ausschnitt aus einer CloudFormation Service-IaC-Datei. Der `environment.outputs.` Namespace bezieht sich auf Ausgaben aus der IaC-Umgebungsdatei. Der `service_instance.inputs.` Namespace bezieht sich auf Eingabeparameter der Dienstinstanz. Die `service_instance.name` Eigenschaft bezieht sich auf einen AWS Proton Ressourcenparameter.

```
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
```

Parameterdetails und Beispiele für die CloudFormation IaC-Komponentendatei

Sie können Parameter in Ihrer Komponenteninfrastruktur als Codedateien (IaC) definieren und referenzieren. Eine ausführliche Beschreibung der AWS Proton Parameter, Parametertypen, des Parameter-Namespace und der Verwendung von Parametern in Ihren IaC-Dateien finden Sie unter. the section called “Parameters” Weitere Informationen zu Komponenten finden Sie unter. Komponenten

Definieren Sie Ausgabeparameter für Komponenten

Sie können Ausgabeparameter in Ihren IaC-Komponentendateien definieren. Sie können dann in Service-IaC-Dateien auf diese Ausgaben verweisen.

> **Note**
>
> Sie können keine Eingaben für Komponenten-IaC-Dateien definieren. Angehängte Komponenten können Eingaben von der Dienstinstanz erhalten, an die sie angehängt sind. Getrennte Komponenten haben keine Eingaben.

Lesen Sie Parameterwerte in IaC-Dateien von Komponenten

Sie können Parameter lesen, die sich auf die Komponente und andere Ressourcen beziehen, in Komponenten-IaC-Dateien. Sie lesen einen Parameterwert, indem Sie im Parameter-Namespace auf den Namen des AWS Proton Parameters verweisen.

- Eingabeparameter — Lesen Sie den Eingabewert einer angehängten Dienstinstanz, indem Sie darauf verweisen. `service_instance.inputs.`*`input-name`*
- Ressourcenparameter — Lesen Sie AWS Proton Ressourcenparameter, indem Sie auf Namen wie`component.name`, `service.name`service_instance.name, und verweisen. `environment.name`
- Ausgabeparameter — Lesen Sie die Umgebungsausgaben durch `environment.outputs.`*`output-name`* Referenzierung.

IaC-Beispieldateien für Komponenten und Dienste mit Parametern

Das folgende Beispiel zeigt eine Komponente, die einen Amazon Simple Storage Service (Amazon S3) -Bucket und zugehörige Zugriffsrichtlinien bereitstellt und die Amazon-Ressourcennamen (ARNs) beider Ressourcen als Komponentenausgaben verfügbar macht. Eine Service-IaC-Vorlage fügt die Komponentenausgaben als Container-Umgebungsvariablen einer Amazon Elastic Container Service (Amazon ECS) -Aufgabe hinzu, um die Ausgaben für Code verfügbar zu machen, der im Container ausgeführt wird, und fügt der Rolle der Aufgabe die Bucket-Zugriffsrichtlinie hinzu. Der Bucket-Name basiert auf den Namen der Umgebung, des Dienstes, der Service-Instance und der Komponente, was bedeutet, dass der Bucket mit einer bestimmten Instance der Komponentenvorlage verknüpft ist, die eine bestimmte Service-Instance erweitert. Entwickler können auf der Grundlage dieser Komponentenvorlage mehrere benutzerdefinierte Komponenten erstellen, um Amazon S3 S3-Buckets für verschiedene Service-Instances und funktionale Anforderungen bereitzustellen.

Das Beispiel zeigt, wie Sie die `{{ ... }}` Jinja-Syntax verwenden, um auf Komponenten- und andere Ressourcenparameter in Ihrer Service-IaC-Datei zu verweisen. Sie können `{% if ... %}` Anweisungen nur dann verwenden, um Anweisungsblöcke hinzuzufügen, wenn eine Komponente an die Dienstinstanz angehängt ist. Bei den `proton_cfn_*` Schlüsselwörtern handelt es sich um Filter, mit denen Sie die Werte der Ausgabeparameter bereinigen und formatieren können. Weitere Informationen zu Filtern finden Sie unter the section called “CloudFormation Parameterfilter”.

Als Administrator erstellen Sie die IaC-Vorlagendatei für den Dienst.

Example CloudFormation Service-IaC-Datei mithilfe einer Komponente

```
# 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:
      # ...
```

Als Entwickler erstellen Sie die IaC-Vorlagendatei für Komponenten.

Example CloudFormation Komponenten-IaC-Datei

```
# 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
```

Wenn eine CloudFormation Vorlage für Ihre Dienstinstanz AWS Proton gerendert und alle Parameter durch tatsächliche Werte ersetzt werden, sieht die Vorlage möglicherweise wie die folgende Datei aus.

Example Die Dienstinstanz hat eine CloudFormation IAC-Datei gerendert

```
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:
      # ...
```

Parameterfilter für CloudFormation IaC-Dateien

Wenn Sie in Ihren AWS CloudFormation IaC-Dateien auf AWS Proton Parameter verweisen, können Sie Jinja-Modifikatoren, sogenannte Filter, verwenden, um Parameterwerte zu überprüfen, zu filtern und zu formatieren, bevor sie in die gerenderte Vorlage eingefügt werden. Filtervalidierungen sind besonders nützlich, wenn auf Ausgabeparameter von Komponenten verwiesen wird, da die Erstellung und das Anhängen von Komponenten von Entwicklern vorgenommen werden und ein Administrator, der Komponentenausgaben in einer Service-Instanzvorlage verwendet, möglicherweise deren Existenz und Gültigkeit überprüfen möchte. Sie können jedoch Filter in jeder Jinja-IaC-Datei verwenden.

In den folgenden Abschnitten werden die verfügbaren Parameterfilter beschrieben und definiert und es werden Beispiele bereitgestellt. AWS Proton definiert die meisten dieser Filter. Der `default` Filter ist ein eingebauter Jinja-Filter.

Umgebungseigenschaften für Amazon ECS-Aufgaben formatieren

Erklärung

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

Beschreibung

Dieser Filter formatiert eine Liste von Ausgaben, die in einer Environment-Eigenschaft im `ContainerDefinition` Abschnitt einer Amazon Elastic Container Service (Amazon ECS) - Aufgabendefinition verwendet werden sollen.

Wird `raw` auf gesetzt`False`, um auch den Parameterwert zu überprüfen. In diesem Fall muss der Wert mit dem regulären Ausdruck übereinstimmen`^[a-zA-Z0-9_-]*$`. Wenn der Wert diese Überprüfung nicht besteht, schlägt das Rendern der Vorlage fehl.

Beispiel

Mit der folgenden benutzerdefinierten Komponentenvorlage:

```
Resources:
  # ...
Outputs:
  Output1:
    Description: "Example component output 1"
```

```
    Value: hello
  Output2:
    Description: "Example component output 2"
    Value: world
```

Und die folgende Servicevorlage:

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

Die Vorlage für den gerenderten Dienst sieht wie folgt aus:

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

Formatieren Sie Umgebungseigenschaften für Lambda-Funktionen

Deklaration

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

Beschreibung

Dieser Filter formatiert eine Liste von Ausgaben, die in einer Environment-Eigenschaft im `Properties` Abschnitt einer AWS Lambda Funktionsdefinition verwendet werden sollen.

Wird `raw` auf gesetzt`False`, um auch den Parameterwert zu überprüfen. In diesem Fall muss der Wert mit dem regulären Ausdruck übereinstimmen`^[a-zA-Z0-9_-]*$`. Wenn der Wert diese Überprüfung nicht besteht, schlägt das Rendern der Vorlage fehl.

Beispiel

Mit der folgenden benutzerdefinierten Komponentenvorlage:

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

Und die folgende Servicevorlage:

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

Die Vorlage für den gerenderten Dienst sieht wie folgt aus:

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

Extrahieren Sie die IAM-Richtlinie ARNs , um sie in IAM-Rollen aufzunehmen

Erklärung

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

Beschreibung

Dieser Filter formatiert eine Liste von Ausgaben, die in einer ManagedPolicyArns Eigenschaft im `Properties` Abschnitt einer AWS Identity and Access Management (IAM-) Rollendefinition verwendet werden sollen. Der Filter verwendet den regulären Ausdruck`^arn:[a-zA-Z-]+:iam::\d{12}:policy/`, um eine gültige IAM-Richtlinie ARNs aus der Liste der Ausgabeparameter zu extrahieren. Sie können diesen Filter verwenden, um Richtlinien in Ausgabeparameterwerten an eine IAM-Rollendefinition in einer Dienstvorlage anzuhängen.

Beispiel

Mit der folgenden benutzerdefinierten Komponentenvorlage:

```
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
```

Und die folgende Servicevorlage:

```
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:
      # ...
```

Die Vorlage für den gerenderten Dienst sieht wie folgt aus:

```
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:
```

```
      # ...
```

Desinfizieren Sie die Immobilienwerte

Erklärung

```
string # proton_cfn_sanitize # string
```

Beschreibung

Dies ist ein Allzweckfilter. Verwenden Sie ihn, um die Sicherheit eines Parameterwerts zu überprüfen. Der Filter überprüft, ob der Wert entweder dem regulären Ausdruck entspricht `^[a-zA-Z0-9_-]*$` oder ob es sich um einen gültigen Amazon-Ressourcennamen (ARN) handelt. Wenn der Wert diese Überprüfung nicht besteht, schlägt das Rendern der Vorlage fehl.

Beispiel

Mit der folgenden benutzerdefinierten Komponentenvorlage:

```
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
```

- Die folgende Referenz in einer Dienstvorlage:

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

  Wird wie folgt gerendert:

  ```
  # ...
  ```

```
  This-is_valid_37
```

- Die folgende Referenz in einer Dienstvorlage:

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

  Ergebnisse mit dem folgenden Renderfehler:

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

- Die folgende Referenz in einer Dienstvorlage:

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

  Wird wie folgt gerendert:

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

Geben Sie Standardwerte für nicht existierende Verweise an

Beschreibung

Der `default` Filter stellt einen Standardwert bereit, wenn kein Namespace-Verweis vorhanden ist. Verwenden Sie ihn, um robuste Vorlagen zu schreiben, die auch dann fehlerfrei gerendert werden können, wenn der Parameter, auf den Sie verweisen, fehlt.

Beispiel

Der folgende Verweis in einer Dienstvorlage führt dazu, dass das Rendern von Vorlagen fehlschlägt, wenn der Dienstinstanz keine direkt definierte (Standard-) Komponente angehängt ist oder wenn die angehängte Komponente keine Ausgabe mit dem Namen hat`test`.

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

Um dieses Problem zu vermeiden, fügen Sie den `default` Filter hinzu.

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

CodeBuild Details und Beispiele für Bereitstellungsparameter

Sie können Parameter in Ihren Vorlagen für CodeBuild basierte AWS Proton Ressourcen definieren und in Ihrem Bereitstellungscode auf diese Parameter verweisen. Eine ausführliche Beschreibung der AWS Proton Parameter, Parametertypen, des Parameter-Namespace und der Verwendung von Parametern in Ihren IaC-Dateien finden Sie unter. the section called “Parameters”

> **Note**
>
> Sie können die CodeBuild Bereitstellung mit Umgebungen und Diensten verwenden. Derzeit können Sie Komponenten auf diese Weise nicht bereitstellen.

Eingabeparameter

Wenn Sie eine AWS Proton Ressource erstellen, z. B. eine Umgebung oder einen Dienst, geben Sie Werte für Eingabeparameter an, die in der Schemadatei Ihrer Vorlage definiert sind. Wenn die von Ihnen erstellte Ressource diese Eingabewerte verwendetCodeBuild Bereitstellung, AWS Proton rendert sie in einer Eingabedatei. Ihr Bereitstellungscode kann Parameterwerte aus dieser Datei importieren und abrufen.

Ein Beispiel für CodeBuild Vorlagen finden Sie unterthe section called “CodeBuild bündeln”. Weitere Informationen zu Manifestdateien finden Sie unter the section called “Manifestieren und abschließen”.

Das folgende Beispiel ist eine JSON-Eingabedatei, die während der CodeBuild basierten Bereitstellung einer Dienstinstanz generiert wurde.

Beispiel: Verwendung von AWS CDK with provisioning CodeBuild

```
{
  "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"
    }
  }
}
```

Ausgabeparameter

Um die Ausgaben der Ressourcenbereitstellung zurück an zu übermitteln AWS Proton, kann Ihr Bereitstellungscode eine JSON-Datei `proton-outputs.json` mit Werten für Ausgabeparameter generieren, die in der Schemadatei Ihrer Vorlage definiert sind. Der `cdk deploy` Befehl hat beispielsweise das `--outputs-file` Argument, das die anweist, eine JSON-Datei mit Bereitstellungsausgaben AWS CDK zu generieren. Wenn Ihre Ressource den verwendet AWS CDK, geben Sie den folgenden Befehl in Ihrem CodeBuild Vorlagenmanifest an:

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

AWS Proton sucht nach dieser JSON-Datei. Wenn die Datei existiert, nachdem Ihr Bereitstellungscode erfolgreich abgeschlossen wurde, AWS Proton liest die Ausgabeparameterwerte aus ihr.

Details und Beispiele für die Terraform-Infrastruktur-as-Code-Datei (IaC)

Sie können Terraform-Eingabevariablen in `variable.tf` Dateien in Ihrem Vorlagenpaket aufnehmen. Sie können auch ein Schema erstellen, um AWS Proton verwaltete Variablen zu erstellen. AWS Proton erstellt eine Variable `.tf files` aus Ihrer Schemadatei. Weitere Informationen finden Sie unter the section called “Terraform-IaC-Dateien”.

Um auf Ihre vom Schema definierten AWS Proton Variablen in Ihrer Infrastruktur zu verweisen `.tf files`, verwenden Sie die AWS Proton Namespaces, die in der Tabelle Parameter und Namespaces für Terraform IaC aufgeführt sind. Sie können beispielsweise die Datei `var.environment.inputs.vpc_cidr` verwenden. Umschließen Sie diese Variablen innerhalb

von Anführungszeichen mit einfachen Klammern und fügen Sie vor der ersten Klammer ein Dollarzeichen hinzu (z. B.). `“${var.environment.inputs.vpc_cidr}”`

Das folgende Beispiel zeigt, wie Namespaces verwendet werden, um AWS Proton Parameter in eine Umgebung einzubeziehen. `.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 als Codedateien

Die Hauptbestandteile des Vorlagenpakets sind IaC-Dateien (Infrastructure as Code), die die Infrastrukturressourcen und Eigenschaften definieren, die Sie bereitstellen möchten. AWS

CloudFormation und andere Infrastruktur-as-Code-Engines verwenden diese Arten von Dateien, um Infrastrukturressourcen bereitzustellen.

> **Note**
>
> Eine IaC-Datei kann auch unabhängig von Vorlagenpaketen als direkte Eingabe für direkt definierte Komponenten verwendet werden. Weitere Informationen zu Komponenten finden Sie unter. Komponenten

AWS Proton unterstützt derzeit zwei Typen von IaC-Dateien:

- CloudFormationDateien — Wird für die AWS verwaltete Bereitstellung verwendet. AWS Proton verwendet Jinja zusätzlich zum CloudFormation Vorlagendateiformat für die Parametrisierung.
- Terraform-HCL-Dateien — Werden für die selbstverwaltete Bereitstellung verwendet. HCL unterstützt nativ die Parametrisierung.

Sie können AWS Proton Ressourcen nicht mit einer Kombination von Bereitstellungsmethoden bereitstellen. Sie müssen die eine oder die andere verwenden. Sie können einen AWS-verwalteten Bereitstellungsdienst nicht in einer selbstverwalteten Bereitstellungsumgebung bereitstellen oder umgekehrt.

Weitere Informationen dazu finden Sie unter the section called “Bereitstellungsmethoden”, Umgebungen, Dienstleistungen und Komponenten.

CloudFormation IaC-Dateien

Erfahren Sie, wie Sie AWS CloudFormation Infrastructure-as-Code-Dateien mit verwenden können. AWS Proton CloudFormation ist ein Infrastructure-as-Code-Service (IaC), der Sie bei der Modellierung und Einrichtung Ihrer AWS Ressourcen unterstützt. Sie definieren Ihre Infrastrukturressourcen in Vorlagen und verwenden Jinja zusätzlich zum CloudFormation Vorlagendateiformat für die Parametrisierung. AWS Proton erweitert die Parameter und rendert die vollständige Vorlage. CloudFormation CloudFormation stellt die definierten Ressourcen als CloudFormation Stapel bereit. Weitere Informationen finden Sie unter Was ist CloudFormation im CloudFormation Benutzerhandbuch enthalten.

AWS Proton unterstützt AWS-verwaltetes Provisioning für CloudFormation IaC.

Beginnen Sie mit Ihrer eigenen vorhandenen Infrastruktur als Codedateien

Sie können Ihre eigene bestehende Infrastruktur als Codedateien (IaC) zur Verwendung mit AWS Proton anpassen.

Die folgenden CloudFormation Beispiele, Beispiel 1 und Beispiel 2, stellen Ihre eigenen vorhandenen CloudFormation IaC-Dateien dar. CloudFormation kann diese Dateien verwenden, um zwei verschiedene CloudFormation Stapel zu erstellen.

In Beispiel 1 ist die CloudFormation IaC-Datei so konfiguriert, dass sie eine Infrastruktur bereitstellt, die von Containeranwendungen gemeinsam genutzt werden kann. In diesem Beispiel werden Eingabeparameter hinzugefügt, sodass Sie dieselbe IaC-Datei verwenden können, um mehrere Sätze bereitgestellter Infrastruktur zu erstellen. Jeder Satz kann unterschiedliche Namen sowie einen anderen Satz von VPC- und Subnetz-CIDR-Werten haben. Als Administrator oder Entwickler geben Sie Werte für diese Parameter an, wenn Sie eine IaC-Datei zur Bereitstellung von Infrastrukturressourcen verwenden. CloudFormation Der Einfachheit halber sind diese Eingabeparameter mit Kommentaren gekennzeichnet und im Beispiel mehrfach referenziert. Die Ausgaben werden am Ende der Vorlage definiert. Sie können in anderen CloudFormation IaC-Dateien referenziert werden.

In Beispiel 2 ist die CloudFormation IaC-Datei so konfiguriert, dass sie eine Anwendung in der Infrastruktur bereitstellt, die in Beispiel 1 bereitgestellt wurde. Die Parameter sind der Einfachheit halber kommentiert.

Beispiel 1: CloudFormation IaC-Datei

```
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"
```

Beispiel 2: CloudFormation IaC-Datei

```
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}"
```

Sie können diese Dateien für die Verwendung mit AWS Proton anpassen.

Bringen Sie Ihre Infrastruktur als Code zu AWS Proton

Mit geringfügigen Änderungen können Sie Beispiel 1 als IaC-Datei (Infrastructure as Code) für ein Umgebungsvorlagenpaket verwenden, das zur Bereitstellung einer Umgebung AWS Proton verwendet wird (wie in Beispiel 3 gezeigt).

Anstatt die CloudFormation Parameter zu verwenden, verwenden Sie die Jinja-Syntax, um auf Parameter zu verweisen, die Sie in einer Open API-basierten Schemadatei definiert haben. Diese Eingabeparameter werden der Einfachheit halber kommentiert und in der IaC-Datei mehrfach referenziert. Auf diese Weise AWS Proton können Parameterwerte geprüft und überprüft werden. Es kann auch Ausgabeparameterwerte in einer IaC-Datei mit Parametern in einer anderen IaC-Datei abgleichen und einfügen.

Als Administrator können Sie den AWS Proton `environment.inputs.` Namespace zu den Eingabeparametern hinzufügen. Wenn Sie in einer Service-IaC-Datei auf die Ausgaben der Umgebungs-IaC-Datei verweisen, können Sie den Ausgaben den `environment.outputs.` Namespace hinzufügen (z. B.). `environment.outputs.ClusterName` Zuletzt umgeben Sie sie mit geschweiften Klammern und Anführungszeichen.

Mit diesen Änderungen können Ihre CloudFormation IaC-Dateien von verwendet werden. AWS Proton

Beispiel 3: AWS Proton Umgebungsinfrastruktur als Codedatei

```
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'
```

Die IaC-Dateien in Beispiel 1 und Beispiel 3 erzeugen leicht unterschiedliche CloudFormation Stapel. Parameter werden in den Stack-Vorlagendateien unterschiedlich angezeigt. In der

CloudFormation Stack-Vorlagendatei von Beispiel 1 werden die Parameterbeschriftungen (Schlüssel) in der Stapelvorlagenansicht angezeigt. In der Vorlagendatei für den AWS Proton CloudFormation Infrastruktur-Stack in Beispiel 3 werden die Parameterwerte angezeigt. AWS Proton Eingabeparameter werden in der Parameteransicht des CloudFormation Konsolen-Stacks nicht angezeigt.

In Beispiel 4 entspricht die AWS Proton Service-IaC-Datei Beispiel 2.

Beispiel 4: IaC-Datei der AWS Proton Dienstinstanz

```
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}"
```

In Beispiel 5 stellt die AWS Proton Pipeline-IaC-Datei die Pipeline-Infrastruktur zur Unterstützung der in Beispiel 4 bereitgestellten Dienstinstanzen bereit.

Beispiel 5: IaC-Datei AWS Proton für die Service-Pipeline

```
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 Vorlagenpaket für die Bereitstellung

Anstatt IaC-Vorlagen zum Rendern von IaC-Dateien zu verwenden und sie mit einer CodeBuild IaC-Provisioning-Engine auszuführen, AWS Proton werden beim Provisioning einfach Ihre Shell-Befehle ausgeführt. Erstellen Sie dazu ein AWS CodeBuild Projekt für die Umgebung im Umgebungskonto und starten Sie einen Job, um Ihre Befehle für jede AWS Proton Ressourcenerstellung oder Aktualisierung auszuführen. AWS Proton Wenn Sie ein Vorlagenpaket erstellen, stellen Sie ein Manifest bereit, das die Befehle zur Bereitstellung und Deprovisionierung der Infrastruktur sowie alle Programme, Skripts und anderen Dateien angibt, die für diese Befehle möglicherweise erforderlich

sind. Ihre Befehle können Eingaben lesen, die die Infrastruktur bereitstellen, und sind für die AWS Proton Bereitstellung oder Deprovisionierung der Infrastruktur und die Generierung von Ausgabewerten verantwortlich.

Das Manifest legt auch fest, wie die Eingabedatei gerendert AWS Proton werden soll, aus der Ihr Code Eingabewerte eingeben und aus der Sie Eingabewerte abrufen können. Es kann in JSON oder HCL gerendert werden. Weitere Informationen über die Eingabeparameter finden Sie unter the section called “CodeBuild Bereitstellungsparameter”. Weitere Informationen zu Manifestdateien finden Sie unter the section called “Manifestieren und abschließen”.

> **Note**
>
> Sie können die CodeBuild Bereitstellung mit Umgebungen und Diensten verwenden. Derzeit können Sie Komponenten auf diese Weise nicht bereitstellen.

Beispiel: Verwendung von AWS CDK with CodeBuild Provisioning

Als Beispiel für die Verwendung von CodeBuild Provisioning können Sie Code hinzufügen, der AWS Cloud Development Kit (AWS CDK) AWS Ressourcen bereitstellt (bereitstellt) und deprovisioniert (zerstört), sowie ein Manifest, das das CDK installiert und Ihren CDK-Code ausführt.

In den folgenden Abschnitten sind Beispieldateien aufgeführt, die Sie in ein CodeBuild Provisioning-Vorlagenpaket aufnehmen können, das eine Umgebung mithilfe von bereitstellt. AWS CDK

Manifest

Die folgende Manifestdatei spezifiziert die CodeBuild Bereitstellung und enthält die Befehle, die für die Installation und Verwendung der Datei AWS CDK, die Verarbeitung der Ausgabedatei und die Rückmeldung der Ausgaben an erforderlich sind. AWS Proton

Example infrastructure/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 }}"
```

Schema

Die folgende Schemadatei definiert Parameter für die Umgebung. Ihr AWS CDK Code kann während der Bereitstellung auf Werte dieser Parameter verweisen.

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 dateien

Die folgenden Dateien sind ein Beispiel für ein Node.js CDK-Projekt.

Example infrastructure/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`
    });
  }
}
```

Gerenderte Eingabedatei

Wenn Sie eine Umgebung mithilfe einer CodeBuild basierten Bereitstellungsvorlage erstellen, wird eine Eingabedatei mit den von Ihnen angegebenen Eingabeparameterwerten AWS Proton gerendert. Ihr Code kann sich auf diese Werte beziehen. Die folgende Datei ist ein Beispiel für eine gerenderte Eingabedatei.

Example infrastructure/proton-inputs.json

```
{
  "environment": {
    "name": "myenv",
    "inputs": {
      "my_sample_input": "10.0.0.0/16",
      "my_other_sample_input": "11.0.0.0/16"
    }
  }
}
```

Terraform-IaC-Dateien

Erfahren Sie, wie Sie Terraform-Infrastruktur-as-Code-Dateien (IaC) mit verwenden. AWS ProtonTerraform ist eine weit verbreitete Open-Source-IaC-Engine, die von entwickelt wurde. HashiCorp Terraform-Module wurden in HashiCorp der HCL-Sprache entwickelt und unterstützen mehrere Anbieter von Backend-Infrastrukturen, darunter Amazon Web Services.

AWS Proton unterstützt die selbstverwaltete Bereitstellung für Terraform IaC.

Ein vollständiges Beispiel für ein Provisioning-Repository, das auf Pull-Anfragen reagiert und die Infrastrukturbereitstellung implementiert, finden Sie in der Automatisierungsvorlage Terraform Actions für on. OpenSource GitHub AWS Proton GitHub

So funktioniert die selbstverwaltete Bereitstellung mit Terraform IaC-Vorlagenpaketdateien:

1. Wenn Sie eine Umgebung aus Terraform-Vorlagenpaketen erstellen, AWS Proton kompiliert Ihre Dateien mit Konsolen- oder Eingabeparametern. `.tf spec file`
2. Es stellt eine Pull-Anfrage, um die kompilierten IaC-Dateien mit dem Repository zusammenzuführen, bei dem Sie sich registriert haben. AWS Proton
3. Wenn die Anfrage genehmigt AWS Proton wurde, wird auf den von Ihnen angegebenen Bereitstellungsstatus gewartet.
4. Wenn die Anfrage abgelehnt wird, wird die Erstellung der Umgebung abgebrochen.
5. Wenn bei der Pull-Anfrage ein Timeout auftritt, ist die Erstellung der Umgebung nicht abgeschlossen.

AWS Proton mit Überlegungen zu Terraform IaC:

- AWS Proton verwaltet Ihre Terraform-Bereitstellung nicht.
- Sie müssen ein Provisioning-Repository bei registrieren. AWS Proton AWS Proton stellt Pull-Requests für dieses Repository.
- Sie müssen eine Verbindung herstellen AWS Proton , um eine CodeStar Verbindung zu Ihrem Provisioning-Repository herzustellen.
- Um Daten aus AWS Proton kompilierten IaC-Dateien bereitzustellen, müssen Sie auf AWS Proton Pull-Requests antworten. AWS Proton sendet Pull-Requests, nachdem die Umgebung und der Dienst Aktionen erstellt und aktualisiert haben. Weitere Informationen erhalten Sie unter AWS Proton Umgebungen und AWS Proton Dienstleistungen.
- Um eine Pipeline aus AWS Proton kompilierten IaC-Dateien bereitzustellen, müssen Sie ein CI/CD Pipeline-Repository erstellen.
- Ihre auf Pull-Requests basierende Bereitstellungsautomatisierung muss Schritte zur Benachrichtigung über alle AWS Proton Statusänderungen der bereitgestellten AWS Proton Ressourcen enthalten. Sie können die API verwenden. AWS Proton NotifyResourceDeploymentStatusChange
- Sie können keine Dienste, Pipelines und Komponenten, die aus CloudFormation IaC-Dateien erstellt wurden, in Umgebungen bereitstellen, die aus Terraform-IaC-Dateien erstellt wurden.

- Sie können keine Dienste, Pipelines und Komponenten, die aus Terraform-IaC-Dateien erstellt wurden, in Umgebungen bereitstellen, die aus IaC-Dateien erstellt wurden. CloudFormation

Bei der Vorbereitung Ihrer Terraform-IaC-Dateien für fügen Sie Ihren Eingabevariablen Namespaces hinzu AWS Proton, wie in den folgenden Beispielen gezeigt. Weitere Informationen finden Sie unter Parameter.

Beispiel 1: Terraform-IaC-Umgebungsdatei AWS Proton

```
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
}
```

Kompilierte Infrastruktur als Code

Wenn Sie eine Umgebung oder einen Dienst erstellen, AWS Proton kompiliert Ihre Infrastruktur als Codedateien mit Konsole oder `spec file` Eingaben. Es erstellt `proton.`*`resource-type`*`.variables.tf` `proton.auto.tfvars.json` Dateien für Ihre Eingaben, die von Terraform verwendet werden können, wie in den folgenden Beispielen gezeigt. Diese Dateien befinden sich in einem angegebenen Repository in einem Ordner, der dem Namen der Umgebung oder der Dienstinstanz entspricht.

Das Beispiel zeigt, wie AWS Proton Tags in die Variablendefinition und Variablenwerte aufgenommen werden und wie Sie diese AWS Proton Tags an bereitgestellte Ressourcen weitergeben können. Weitere Informationen finden Sie unter the section called “Weitergabe von Tags an bereitgestellte Ressourcen”.

Beispiel 2: kompilierte IAc-Dateien für eine Umgebung mit dem Namen „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"
  }
}
```

Repository-Pfade

AWS Proton verwendet Konsolen- oder Spezifikationseingaben aus Aktionen zur Erstellung von Umgebungen oder Diensten, um das Repository und den Pfad zu finden, in dem die kompilierten IaC-Dateien gespeichert werden sollen. Die Eingabewerte werden an Eingabeparameter mit Namespaces übergeben.

AWS Proton unterstützt zwei Layouts für Repository-Pfade. In den folgenden Beispielen werden die Pfade anhand der Namespace-Ressourcenparameter aus zwei Umgebungen benannt. Jede Umgebung hat Dienstinstanzen von zwei Diensten, und die Dienstinstanzen eines der Dienste haben direkt definierte Komponenten.

Ressourcentyp	Benennen Sie den Parameter	=	Ressourcenname
Umgebung	`environment.name`		`"env-prod"`
Umgebung	`environment.name`		`"env-staged"`
Service	`service.name`		`"service-one"`
Service-Instance	`service_instance.name`	=	`"instance-one-prod"`
Service-Instance	`service_instance.name`		`"instance-one-staged"`
Service	`service.name`		`"service-two"`
Service-Instance	`service_instance.name`		`"instance-two-prod"`

Ressourcentyp	Benennen Sie den Parameter	=	Ressourcennname
Komponente	`service_instance.components.default.name`		`"component-prod"`
Service-Instance	`service_instance.name`		`"instance-two-staged"`
Komponente	`service_instance.components.default.name`		`"component-staged"`

Layout 1

Wenn AWS Proton das angegebene Repository mit einem `environments` Ordner gefunden wird, erstellt es einen Ordner, der die kompilierten IaC-Dateien enthält und mit dem `environment.name` benannt ist.

Wenn das AWS Proton angegebene Repository mit einem `environments` Ordner gefunden wird, der einen Ordnernamen enthält, der einem mit einer Dienstinstanz kompatiblen Umgebungsnamen entspricht, erstellt es einen Ordner, der die kompilierten Instanz-IaC-Dateien enthält und mit dem benannt ist. `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

Wenn das AWS Proton angegebene Repository ohne Ordner gefunden wird, erstellt es einen `environments` Ordner, in dem sich die kompilierten IaC-Umgebungsdateien `environment.name` befinden.

Wenn das AWS Proton angegebene Repository mit einem Ordnernamen gefunden wird, der einem mit einer Dienstinstanz kompatiblen Umgebungsnamen entspricht, erstellt es einen `service_instance.name` Ordner, in dem die kompilierten Instanz-IaC-Dateien gespeichert werden.

```
/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
```

Schemadatei

Wenn Sie als Administrator den Abschnitt „Offene API-Datenmodelle (Schemas)“ verwenden, um eine Parameterschema-YAML-Datei für Ihr Vorlagenpaket zu definieren, AWS Proton können Sie

Parameterwerteingaben anhand der Anforderungen überprüfen, die Sie in Ihrem Schema definiert haben.

Weitere Informationen zu Formaten und verfügbaren Schlüsselwörtern finden Sie im Abschnitt Schema-Objekt der OpenAPI.

Schemaanforderungen für Umgebungsvorlagenpakete

Ihr Schema muss dem Abschnitt Datenmodelle (Schemas) der OpenAPI im YAML-Format entsprechen. Es muss auch Teil Ihres Umgebungsvorlagenpakets sein.

Für Ihr Umgebungsschema müssen Sie die formatierten Header angeben, um sicherzustellen, dass Sie den Abschnitt Datenmodelle (Schemas) der Open API verwenden. In den folgenden Beispielen für Umgebungsschemas erscheinen diese Header in den ersten drei Zeilen.

Ein `environment_input_type` muss enthalten und mit einem von Ihnen angegebenen Namen definiert werden. In den folgenden Beispielen ist dies in Zeile 5 definiert. Indem Sie diesen Parameter definieren, verknüpfen Sie ihn mit einer AWS Proton Umgebungsressource.

Um dem Open-API-Schemamodell zu folgen, müssen Sie Folgendes einbeziehen`types`. Im folgenden Beispiel ist dies Zeile 6.

`types`Im Folgenden müssen Sie einen `environment_input_type` Typ definieren. Sie definieren die Eingabeparameter für Ihre Umgebung als Eigenschaften von`environment_input_type`. Sie müssen mindestens eine Eigenschaft angeben, deren Name mit mindestens einem Parameter übereinstimmt, der in der Umgebungsinfrastrukturdatei als Codedatei (IaC) aufgeführt ist, die dem Schema zugeordnet ist.

Wenn Sie eine Umgebung erstellen und benutzerdefinierte Parameterwerte angeben, AWS Proton verwendet er die Schemadatei, um sie abzugleichen, zu validieren und in die Parameter mit geschweiften Klammern in der zugehörigen IaC-Datei einzufügen. CloudFormation Geben Sie für jede Eigenschaft (Parameter) ein und ein. `name` `type` Geben Sie optional auch ein `description`, `default`, und an`pattern`.

Zu den definierten Parametern für das folgende Beispielschema für eine Standardumgebungsvorlage gehören`vpc_cidr`,`subnet_one_cidr`, und `subnet_two_cidr` zusammen mit dem `default` Schlüsselwort und den Standardwerten. Wenn Sie eine Umgebung mit diesem Umgebungsvorlagen-Bundle-Schema erstellen, können Sie die Standardwerte akzeptieren oder Ihre eigenen angeben. Wenn ein Parameter keinen Standardwert hat und als `required` Eigenschaft (Parameter) aufgeführt ist, müssen Sie beim Erstellen einer Umgebung Werte dafür angeben.

Im zweiten Beispiel eines Schemas für eine Standardumgebungsvorlage ist der `required` Parameter aufgeführt`my_other_sample_input`.

Sie können ein Schema für zwei Arten von Umgebungsvorlagen erstellen. Weitere Informationen finden Sie unter Vorlagen registrieren und veröffentlichen.

- Vorlagen für Standardumgebungen

  Im folgenden Beispiel wird ein Umgebungseingabetyp mit einer Beschreibung und Eingabeeigenschaften definiert. Dieses Schemabeispiel kann mit der in Beispiel 3 gezeigten AWS Proton CloudFormation IaC-Datei verwendet werden.

  Beispielschema für eine Standardumgebungsvorlage:

```
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))
```

  Beispielschema für eine Standardumgebungsvorlage, die einen `required` Parameter enthält:

```
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
```

- Vorlagen für vom Kunden verwaltete Umgebungen

  Im folgenden Beispiel enthält das Schema nur eine Liste von Ausgaben, die die Ausgaben der IaC replizieren, mit der Sie Ihre vom Kunden verwaltete Infrastruktur bereitgestellt haben. Sie müssen Ausgabewerttypen nur als Zeichenfolgen definieren (nicht als Listen, Arrays oder andere Typen). Der nächste Codeausschnitt zeigt beispielsweise den Ausgabebereich einer externen Vorlage. CloudFormation Dies ist aus der in Beispiel 1 gezeigten Vorlage. Es kann verwendet werden, um eine externe, vom Kunden verwaltete Infrastruktur für einen AWS Proton Fargate-Dienst zu erstellen, der aus Beispiel 4 erstellt wurde.

  > **Important**
  >
  > Als Administrator müssen Sie sicherstellen, dass Ihre bereitgestellte und verwaltete Infrastruktur und alle Ausgabeparameter mit den zugehörigen kundenverwalteten Umgebungsvorlagen kompatibel sind. AWS Proton kann Änderungen in Ihrem Namen

nicht berücksichtigen, da diese Änderungen für Sie nicht sichtbar sind. AWS Proton Inkonsistenzen führen zu Fehlern.

Beispiele für CloudFormation IaC-Dateiausgaben für eine vom Kunden verwaltete Umgebungsvorlage:

```
// 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'
[...]
```

Das Schema für das entsprechende Vorlagenpaket für vom AWS Proton Kunden verwaltete Umgebungen wird im folgenden Beispiel gezeigt. Jeder Ausgabewert ist als Zeichenfolge definiert.

Beispielschema für eine vom Kunden verwaltete Umgebungsvorlage:

```
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"
[...]
```

Schemaanforderungen für Servicevorlagenpakete

Ihr Schema muss dem Abschnitt Datenmodelle (Schemas) der OpenAPI im YAML-Format entsprechen, wie in den folgenden Beispielen gezeigt. Sie müssen eine Schemadatei in Ihrem Service-Template-Paket bereitstellen.

In den folgenden Beispielen für ein Serviceschema müssen Sie die formatierten Header einbeziehen. Im folgenden Beispiel ist dies in den ersten drei Zeilen. Damit stellen Sie sicher, dass Sie den Abschnitt Datenmodelle (Schemas) der Open API verwenden.

A `service_input_type` muss enthalten und mit einem von Ihnen angegebenen Namen definiert werden. Im folgenden Beispiel befindet sich dies in Zeile 5. Dadurch werden die Parameter einer AWS Proton Dienstressource zugeordnet.

Eine AWS Proton Service-Pipeline ist standardmäßig enthalten, wenn Sie die Konsole oder die CLI verwenden, um einen Service zu erstellen. Wenn Sie eine Service-Pipeline für Ihren Service hinzufügen, müssen Sie sie `pipeline_input_type` mit einem von Ihnen angegebenen Namen angeben. Im folgenden Beispiel ist dies in Zeile 7. Fügen Sie diesen Parameter nicht ein, wenn Sie keine AWS Proton Service-Pipeline einbeziehen. Weitere Informationen finden Sie unter Vorlagen registrieren und veröffentlichen.

Um dem Open-API-Schemamodell zu folgen, müssen Sie Folgendes angeben: Im folgenden Beispiel steht dies in Zeile 9. `types`

`types`Im Folgenden müssen Sie einen `service_input_type` Typ definieren. Sie definieren die Eingabeparameter für Ihren Service als Eigenschaften von`service_input_type`. Sie müssen mindestens eine Eigenschaft mit einem Namen angeben, der mit mindestens einem Parameter übereinstimmt, der in der Datei Service Infrastructure as Code (IaC) aufgeführt ist, die dem Schema zugeordnet ist.

Um eine Service-Pipeline zu definieren, müssen Sie unterhalb Ihrer `service_input_type` Definition eine `pipeline_input_type` definieren. Wie oben müssen Sie mindestens eine Eigenschaft mit einem Namen angeben, der mit mindestens einem Parameter übereinstimmt, der in

einer Pipeline-IaC-Datei aufgeführt ist, die dem Schema zugeordnet ist. Nehmen Sie diese Definition nicht auf, wenn Sie keine AWS Proton Service-Pipeline einbeziehen.

Wenn Sie als Administrator oder Entwickler einen Service erstellen und benutzerdefinierte Parameterwerte angeben, AWS Proton verwendet er die Schemadatei, um sie abzugleichen, zu validieren und in die Parameter der zugehörigen CloudFormation IaC-Datei mit geschweiften Klammern einzufügen. Geben Sie für jede Eigenschaft (Parameter) a und a an. `name` `type` Geben Sie optional auch ein `description``default`, und an`pattern`.

Zu den definierten Parametern für das Beispielschema gehören `port``desired_count`, `task_size` und `image` zusammen mit dem `default` Schlüsselwort und den Standardwerten. Wenn Sie einen Service mit diesem Dienstvorlagen-Bundle-Schema erstellen, können Sie die Standardwerte akzeptieren oder eigene Werte angeben. Der Parameter `unique_name` ist auch im Beispiel enthalten und hat keinen Standardwert. Er ist als `required` Eigenschaft (Parameter) aufgeführt. Sie als Administrator oder Entwickler müssen Werte für `required` Parameter angeben, wenn Sie Dienste erstellen.

Wenn Sie eine Dienstvorlage mit einer Service-Pipeline erstellen möchten, nehmen Sie die `pipeline_input_type` in Ihr Schema auf.

Beispiel für eine Dienstschemadatei für einen Dienst, der eine AWS Proton Dienstpipeline enthält.

Dieses Schemabeispiel kann mit den in Beispiel 4 und Beispiel 5 gezeigten AWS Proton IaC-Dateien verwendet werden. Eine Service-Pipeline ist enthalten.

```
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
```

Wenn Sie eine Dienstvorlage ohne Service-Pipeline erstellen möchten, nehmen Sie die nicht `pipeline_input_type` in Ihr Schema auf, wie im folgenden Beispiel gezeigt.

Beispiel für eine Dienstschemadatei für einen Service, der keine AWS Proton Service-Pipeline enthält

```
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"
```

Verpacken Sie die Vorlagendateien für AWS Proton

Nachdem Sie Ihre Umgebung und Dienstinfrastruktur als Codedateien (IaC) und die entsprechenden Schemadateien vorbereitet haben, müssen Sie sie in Verzeichnissen organisieren. Sie müssen auch eine Manifest-YAML-Datei erstellen. Die Manifestdatei listet die IaC-Dateien in einem Verzeichnis, die Rendering-Engine und die Vorlagensprache auf, die zur Entwicklung der IaC in dieser Vorlage verwendet wurde.

> **Note**
>
> Eine Manifestdatei kann auch unabhängig von Vorlagenpaketen als direkte Eingabe für direkt definierte Komponenten verwendet werden. In diesem Fall spezifiziert sie immer eine

einzelne IaC-Vorlagendatei, sowohl für Terraform als auch für Terraform CloudFormation . Weitere Informationen zu Komponenten finden Sie unter. Komponenten

Die Manifestdatei muss dem Format und dem Inhalt entsprechen, die im folgenden Beispiel gezeigt werden.

CloudFormation Manifest-Dateiformat:

Mit CloudFormation listen Sie eine einzelne Datei auf.

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

Terraform-Manifest-Dateiformat:

Mit Terraform können Sie explizit eine einzelne Datei auflisten oder den Platzhalter verwenden, * um jede der Dateien in einem Verzeichnis aufzulisten.

> **Note**
>
> Der Platzhalter umfasst nur Dateien, deren Namen mit enden. `.tf` Andere Dateien werden ignoriert.

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

CodeBuildbasiertes Provisioning-Manifest-Dateiformat:

Bei der CodeBuild basierten Bereitstellung geben Sie Shell-Befehle für die Bereitstellung und Deprovisionierung an.

> **Note**
>
> Zusätzlich zum Manifest sollte Ihr Paket alle Dateien enthalten, von denen Ihre Befehle abhängen.

Das folgende Beispielmanifest verwendet die CodeBuild basierte Bereitstellung, um Ressourcen mithilfe von () bereitzustellen (bereitzustellen) und zu deprovisionieren (zu zerstören). AWS Cloud Development Kit (AWS CDK) AWS CDK Das Vorlagenpaket sollte auch den CDK-Code enthalten.

Während der Bereitstellung erstellt AWS Proton eine Eingabedatei mit Werten für Eingabeparameter, die Sie im Schema der Vorlage mit dem Namen definiert haben. `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 }}"
```

Nachdem Sie die Verzeichnisse und Manifestdateien für Ihre Umgebung oder Ihr Service-Template-Paket eingerichtet haben, komprimieren Sie die Verzeichnisse in einen Tar-Ball und laden sie in einen Amazon Simple Storage Service (Amazon S3) -Bucket hoch, wo Sie sie abrufen AWS Proton können, oder in ein Git-Repository zur Vorlagensynchronisierung.

Wenn Sie eine Nebenversion einer Umgebung oder einer Service-Vorlage erstellen, bei der Sie sich registriert haben AWS Proton, geben Sie den Pfad zu Ihrem Umgebungs- oder Service-Vorlagen-Bundle-Tarball an, der sich in Ihrem S3-Bucket befindet. AWS Proton speichert es zusammen mit der neuen Nebenversion der Vorlage. Sie können die neue Nebenversion der Vorlage auswählen, um Umgebungen oder Dienste zu erstellen oder zu aktualisieren AWS Proton.

Zusammenfassung des Pakets mit den Umgebungsvorlagen

Es gibt zwei Arten von Umgebungsvorlagenpaketen, für AWS Proton die Sie erstellen.

- Um ein Umgebungsvorlagenpaket für eine Standardumgebungsvorlage zu erstellen, organisieren Sie das Schema, die Infrastructure-as-Code-Dateien (IaC) und die Manifestdatei in Verzeichnissen, wie in der folgenden Verzeichnisstruktur des Umgebungsvorlagen-Bundles dargestellt.
- Um ein Umgebungsvorlagenpaket für eine vom Kunden verwaltete Umgebungsvorlage zu erstellen, geben Sie nur die Schemadatei und das Verzeichnis an. Schließen Sie das Infrastrukturverzeichnis und die Dateien nicht ein. AWS Proton gibt einen Fehler aus, wenn das Infrastrukturverzeichnis und die Dateien enthalten sind.

Weitere Informationen finden Sie unter Vorlagen registrieren und veröffentlichen.

CloudFormation Verzeichnisstruktur des Umgebungsvorlagenpakets:

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

Verzeichnisstruktur des Paketpakets für Terraform-Umgebungsvorlagen:

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

```
  environment.tf
```

Zusammenfassung des Servicevorlagenpakets

Um ein Servicevorlagenpaket zu erstellen, müssen Sie das Schema, die Infrastructure-as-Code-Dateien (IaC) und die Manifestdateien in Verzeichnissen organisieren, wie im Beispiel für die Verzeichnisstruktur des Service Template-Bundles gezeigt.

Wenn Sie keine Service-Pipeline in Ihr Vorlagenpaket aufnehmen, schließen Sie das Pipeline-Verzeichnis und die Pipeline-Dateien nicht ein und legen Sie das fest, `"pipelineProvisioning": "CUSTOMER_MANAGED"` wenn Sie die Servicevorlage erstellen, die diesem Vorlagenpaket zugeordnet werden soll.

> **Note**
>
> Sie können keine Änderungen vornehmen, `pipelineProvisioning` nachdem die Servicevorlage erstellt wurde.

Weitere Informationen finden Sie unter Vorlagen registrieren und veröffentlichen.

CloudFormation Verzeichnisstruktur des Dienstvorlagenpakets:

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

Verzeichnisstruktur des Terraform-Dienstvorlagenpakets:

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

```
    manifest.yaml
    pipeline.tf
```

Überlegungen zum Vorlagenpaket

- Infrastructure-as-Code-Dateien (IaC)

  AWS Proton prüft Vorlagen auf das richtige Dateiformat. Prüft jedoch AWS Proton nicht auf Vorlagenentwicklungs-, Abhängigkeits- und Logikfehler. Nehmen wir beispielsweise an, dass Sie die Erstellung eines Amazon S3 S3-Buckets in Ihrer CloudFormation IaC-Datei als Teil Ihrer Service- oder Umgebungsvorlage angegeben haben. Auf der Grundlage dieser Vorlagen wird ein Service erstellt. Nehmen wir nun an, Sie möchten den Dienst irgendwann löschen. Wenn der angegebene S3-Bucket nicht leer ist und die CloudFormation IaC-Datei ihn nicht als `Retain` in kennzeichnet`DeletionPolicy`, AWS Proton schlägt der Vorgang zum Löschen des Dienstes fehl.
- Größenbeschränkungen und Format der Bundle-Dateien
  - Die Beschränkungen für Größe, Anzahl und Namensgröße von Bundle-Dateien finden Sie unterAWS Proton Kontingente.
  - Die Dateiverzeichnisse der Template-Bundles werden in einem Tar-Ball gezippt und befinden sich in einem Amazon Simple Storage Service (Amazon S3) -Bucket.
  - Jede Datei im Paket muss eine gültige formatierte YAML-Datei sein.
- Verschlüsselung des S3-Bucket-Vorlagenpakets

  Wenn Sie sensible Daten in Ihren Template-Bundles verschlüsseln möchten, die sich in Ihrem S3-Bucket befinden, verwenden Sie SSE-S3- oder SSE-KMS-Schlüssel, um sie abrufen zu können. AWS Proton

AWS Proton Vorlagen

Um Ihr Vorlagenpaket zu Ihrer AWS Proton Vorlagenbibliothek hinzuzufügen, erstellen Sie eine Template-Nebenversion und registrieren Sie sie bei AWS Proton. Geben Sie bei der Erstellung der Vorlage den Namen des Amazon S3 S3-Buckets und den Pfad für Ihr Vorlagenpaket an. Nachdem die Vorlagen veröffentlicht wurden, können sie von Mitgliedern des Plattformteams und Entwicklern ausgewählt werden. Nachdem sie ausgewählt wurden, AWS Proton verwendet die Vorlage, um Infrastruktur und Anwendungen zu erstellen und bereitzustellen.

Als Administrator können Sie eine Umgebungsvorlage mit erstellen und registrieren AWS Proton. Diese Umgebungsvorlage kann dann verwendet werden, um mehrere Umgebungen bereitzustellen. Sie kann beispielsweise verwendet werden, um „Dev“ -, „Staging“ - und „Prod“ -Umgebungen bereitzustellen. Die Entwicklungsumgebung kann eine VPC mit privaten Subnetzen und einer restriktiven Zugriffsrichtlinie für alle Ressourcen umfassen. Umgebungsausgaben können als Eingaben für Dienste verwendet werden.

Sie können Umgebungsvorlagen erstellen und registrieren, um zwei verschiedene Arten von Umgebungen zu erstellen. Sowohl Sie als auch Entwickler können AWS Proton damit Dienste für beide Typen bereitstellen.

- Registrieren und veröffentlichen Sie eine Standardumgebungsvorlage AWS Proton , mit der eine Standardumgebung erstellt wird, die die Umgebungsinfrastruktur bereitstellt und verwaltet.
- Registrieren und veröffentlichen Sie eine Vorlage für eine vom Kunden verwaltete Umgebung, AWS Proton mit der eine vom Kunden verwaltete Umgebung erstellt wird, die eine Verbindung zu Ihrer vorhandenen bereitgestellten Infrastruktur herstellt. AWS Proton verwaltet Ihre bestehende bereitgestellte Infrastruktur nicht.

Sie können Dienstvorlagen erstellen und registrieren AWS Proton , um Dienste in Umgebungen bereitzustellen. Eine AWS Proton Umgebung muss erstellt werden, bevor ein Dienst in ihr bereitgestellt werden kann.

In der folgenden Liste wird beschrieben, wie Sie Vorlagen mit erstellen und verwalten AWS Proton.

- (Optional) Bereiten Sie eine IAM-Rolle vor, um den Entwicklerzugriff auf AWS Proton API-Aufrufe und AWS Proton IAM-Servicerollen zu kontrollieren. Weitere Informationen finden Sie unter the section called “IAM-Rollen”.
- Stellen Sie ein Vorlagenpaket zusammen. Weitere Informationen finden Sie unter Vorlagenpakete.

- Erstellen und registrieren Sie eine Vorlage mit, AWS Proton nachdem das Vorlagenpaket zusammengestellt, komprimiert und in einem Amazon S3 S3-Bucket gespeichert wurde. Sie können dies entweder in der Konsole oder mit dem tun AWS CLI.
- Testen und verwenden Sie die Vorlage, um AWS Proton bereitgestellte Ressourcen zu erstellen und zu verwalten, nachdem sie registriert wurde. AWS Proton
- Erstellen und verwalten Sie Haupt- und Nebenversionen der Vorlage während der gesamten Lebensdauer der Vorlage.

Sie können Vorlagenversionen manuell oder mit Konfigurationen für die Vorlagensynchronisierung verwalten:

- Verwenden Sie die AWS Proton Konsole und erstellen AWS CLI Sie eine neue Neben- oder Hauptversion.
- Erstellen Sie eine Konfiguration für die Vorlagensynchronisierung, mit der AWS Proton automatisch eine neue Neben- oder Hauptversion erstellt werden kann, wenn eine Änderung an Ihrem Vorlagenpaket in einem von Ihnen definierten Repository erkannt wird.

Weitere Informationen finden Sie in der Referenz zur AWS Proton Service-API.

Topics

Versionierte Vorlagen

Als Administrator oder Mitglied eines Plattformteams definieren, erstellen und verwalten Sie eine Bibliothek mit versionierten Vorlagen, die zur Bereitstellung von Infrastrukturressourcen verwendet werden. Es gibt zwei Arten von Vorlagenversionen: Nebenversionen und Hauptversionen.

- Nebenversionen — Änderungen an der Vorlage, die über ein abwärtskompatibles Schema verfügen. Für diese Änderungen muss der Entwickler bei der Aktualisierung auf die neue Vorlagenversion keine neuen Informationen bereitstellen.

  Wenn Sie versuchen, eine geringfügige Versionsänderung AWS Proton vorzunehmen, versucht nach besten Kräften festzustellen, ob das Schema der neuen Version abwärtskompatibel mit den vorherigen Nebenversionen der Vorlage ist. Wenn das neue Schema nicht abwärtskompatibel ist, AWS Proton schlägt die Registrierung der neuen Nebenversion fehl.

  > **Note**
  >
  > Die Kompatibilität wird ausschließlich anhand des Schemas bestimmt. AWS Proton prüft nicht, ob die IaC-Datei (Infrastructure as Code) des Vorlagenpakets abwärtskompatibel mit den vorherigen Nebenversionen ist. Prüft beispielsweise AWS Proton nicht, ob die neue IaC-Datei grundlegende Änderungen für die Anwendungen verursacht, die auf der Infrastruktur ausgeführt werden, die durch eine frühere Nebenversion der Vorlage bereitgestellt wurde.

- Hauptversionen — Änderungen an der Vorlage, die möglicherweise nicht abwärtskompatibel sind. Diese Änderungen erfordern in der Regel neue Eingaben vom Entwickler und beinhalten häufig Änderungen am Vorlagenschema.

  Manchmal entscheiden Sie sich möglicherweise dafür, eine abwärtskompatible Änderung als Hauptversion zu bezeichnen, die auf dem Betriebsmodell Ihres Teams basiert.

Die Art und Weise, wie AWS Proton bestimmt wird, ob sich eine Anfrage auf eine Vorlagenversion für eine Neben- oder Hauptversion bezieht, hängt davon ab, wie Vorlagenänderungen nachverfolgt werden:

- Wenn Sie explizit eine Anfrage zur Erstellung einer neuen Vorlagenversion stellen, fordern Sie eine Hauptversion an, indem Sie eine Hauptversionsnummer angeben, und Sie fordern eine Nebenversion an, wenn Sie keine Hauptversionsnummer angeben.
- Wenn Sie die Vorlagensynchronisierung verwenden (und daher keine expliziten Anfragen zur Vorlagenversion stellen), wird AWS Proton versucht, neue Nebenversionen für Vorlagenänderungen zu erstellen, die in der vorhandenen YAML-Datei vorgenommen werden. AWS Proton erstellt eine Hauptversion, wenn Sie ein neues Verzeichnis für die neue Vorlagenänderung erstellen (z. B. von Version 1 zu Version 2 wechseln).

> **Note**
>
> Eine Registrierung neuer Nebenversionen auf der Grundlage der Vorlagensynchronisierung schlägt immer noch fehl, AWS Proton wenn festgestellt wird, dass die Änderung nicht abwärtskompatibel ist.

Wenn Sie eine neue Version einer Vorlage veröffentlichen, wird sie zur empfohlenen Version, sofern es sich um die höchste Haupt- und Nebenversion handelt. Neue AWS Proton Ressourcen werden mit der neuen empfohlenen Version erstellt. Administratoren werden AWS Proton aufgefordert, die neue Version zu verwenden und vorhandene AWS Proton Ressourcen zu aktualisieren, die eine veraltete Version verwenden.

Vorlagen registrieren und veröffentlichen

Sie können Umgebungs- und Dienstvorlagen mit registrieren und veröffentlichen AWS Proton, wie in den folgenden Abschnitten beschrieben.

Sie können eine neue Version einer Vorlage mit der Konsole oder erstellen AWS CLI.

Alternativ können Sie die Konsole verwenden oder AWS CLI um eine Vorlage zu erstellen und eine Vorlagensynchronisierung dafür zu konfigurieren und zu konfigurieren. Diese Konfiguration ermöglicht die AWS Proton Synchronisierung anhand von Vorlagenpaketen, die sich in registrierten Git-Repositorys befinden, die Sie definiert haben. Immer wenn ein Commit in dein Repository übertragen wird, das eines deiner Template-Bundles ändert, wird eine neue Neben- oder Hauptversion deines Templates erstellt, sofern die Version noch nicht existiert. Weitere Informationen zu den Voraussetzungen und Anforderungen für die Konfiguration der Template-Sync finden Sie unterKonfigurationen für die Vorlagensynchronisierung.

Registrieren und veröffentlichen Sie Umgebungsvorlagen

Sie können die folgenden Typen von Umgebungsvorlagen registrieren und veröffentlichen.

- Registrieren und veröffentlichen Sie eine Standardumgebungsvorlage, die zur Bereitstellung und Verwaltung der Umgebungsinfrastruktur AWS Proton verwendet wird.
- Registrieren und veröffentlichen Sie eine Vorlage für eine vom Kunden verwaltete Umgebung, die AWS Proton verwendet wird, um eine Verbindung zu Ihrer vorhandenen bereitgestellten

Infrastruktur herzustellen, die Sie verwalten. AWS Proton verwaltet Ihre bestehende bereitgestellte Infrastruktur nicht.

> **Important**
>
> Stellen Sie als Administrator sicher, dass Ihre bereitgestellte und verwaltete Infrastruktur und alle Ausgabeparameter mit den zugehörigen, vom Kunden verwalteten Umgebungsvorlagen kompatibel sind. AWS Proton kann Änderungen in Ihrem Namen nicht berücksichtigen, da diese Änderungen für Sie nicht sichtbar sind. AWS Proton Inkonsistenzen führen zu Fehlern.

Sie können die Konsole oder die verwenden, um eine Umgebungsvorlage AWS CLI zu registrieren und zu veröffentlichen.

AWS-Managementkonsole

Verwenden Sie die Konsole, um eine neue Umgebungsvorlage zu registrieren und zu veröffentlichen.

1. Wählen Sie in der AWS Proton Konsole Umgebungsvorlagen aus.
2. Wählen Sie Umgebungsvorlage erstellen aus.
3. Wählen Sie auf der Seite Umgebungsvorlage erstellen im Abschnitt Vorlagenoptionen eine der beiden verfügbaren Vorlagenoptionen aus.
   - Erstellen Sie eine Vorlage für die Bereitstellung neuer Umgebungen.
   - Erstellen Sie eine Vorlage für die Verwendung der bereitgestellten Infrastruktur, die Sie verwalten.
4. Wenn Sie Vorlage für die Bereitstellung neuer Umgebungen erstellen ausgewählt haben, wählen Sie im Abschnitt Quelle des Vorlagenpakets eine der drei verfügbaren Quelloptionen für das Vorlagenpaket aus. Weitere Informationen zu den Anforderungen und Voraussetzungen für die Synchronisierung von Vorlagen finden Sie unter. Konfigurationen für die Vorlagensynchronisierung
   - Verwenden Sie eines unserer Beispiel-Vorlagenpakete.
   - Verwenden Sie Ihr eigenes Vorlagenpaket.
   - Synchronisieren Sie Vorlagen von Git.

5. Geben Sie einen Pfad zu einem Vorlagenpaket an.

   a. Wenn Sie eines unserer Beispiel-Vorlagenpakete verwenden ausgewählt haben:

      Wählen Sie im Abschnitt Mustervorlagenpaket ein Mustervorlagenpaket aus.

   b. Wenn du Vorlagen von Git synchronisieren ausgewählt hast, gehe im Abschnitt Quellcode wie folgt vor:

      i. Wählen Sie das Repository für Ihre Template-Synchronisierungskonfiguration aus.

      ii. Geben Sie den Namen des Repository-Zweigs ein, von dem aus synchronisiert werden soll.

      iii. (Optional) Geben Sie den Namen eines Verzeichnisses ein, um die Suche nach Ihrem Vorlagenpaket einzuschränken.

   c. Andernfalls geben Sie im Abschnitt Speicherort des S3-Bundles einen Pfad zu Ihrem Vorlagenpaket an.

6. Im Abschnitt Vorlagendetails.

   a. Geben Sie einen Namen für die Vorlage ein.

   b. (Optional) Geben Sie einen Anzeigenamen für die Vorlage ein.

   c. (Optional) Geben Sie eine Vorlagenbeschreibung für die Umgebungsvorlage ein.

7. (Optional) Aktivieren Sie im Abschnitt Verschlüsselungseinstellungen das Kontrollkästchen Verschlüsselungseinstellungen anpassen (erweitert), um Ihren eigenen Verschlüsselungsschlüssel anzugeben.

8. (Optional) Wählen Sie im Abschnitt Tags die Option Neues Tag hinzufügen aus und geben Sie einen Schlüssel und einen Wert ein, um ein vom Kunden verwaltetes Tag zu erstellen.

9. Wählen Sie „Umgebungsvorlage erstellen“.

   Sie befinden sich jetzt auf einer neuen Seite, auf der der Status und die Details für Ihre neue Umgebungsvorlage angezeigt werden. Zu diesen Details gehören eine Liste von AWS und vom Kunden verwaltete Tags. AWS Proton generiert automatisch AWS verwaltete Tags für Sie, wenn Sie AWS Proton Ressourcen erstellen. Weitere Informationen finden Sie unter AWS Proton Ressourcen und Tagging.

10. Der Status einer neuen Umgebungsvorlage beginnt im Status Entwurf. Sie und andere Personen mit `proton:CreateEnvironment` entsprechenden Berechtigungen können sie anzeigen und darauf zugreifen. Folgen Sie dem nächsten Schritt, um die Vorlage für andere verfügbar zu machen.

11. Wählen Sie im Abschnitt Vorlagenversionen das Optionsfeld links neben der Nebenversion der Vorlage, die Sie gerade erstellt haben (1.0). Alternativ können Sie in der Informationswarnung die Option Veröffentlichen auswählen und den nächsten Schritt überspringen.
12. Wählen Sie im Abschnitt Vorlagenversionen die Option Veröffentlichen aus.
13. Der Status der Vorlage ändert sich in „Veröffentlicht“. Da es sich um die neueste Version der Vorlage handelt, handelt es sich um die empfohlene Version.
14. Wählen Sie im Navigationsbereich Umgebungsvorlagen aus, um eine Liste Ihrer Umgebungsvorlagen und Details anzuzeigen.

Verwenden Sie die Konsole, um neue Haupt- und Nebenversionen einer Umgebungsvorlage zu registrieren.

Weitere Informationen finden Sie unter Versionierte Vorlagen.

1. Wählen Sie in der AWS Proton Konsole Environment Templates aus.
2. Wählen Sie in der Liste der Umgebungsvorlagen den Namen der Umgebungsvorlage aus, für die Sie eine Haupt- oder Nebenversion erstellen möchten.
3. Wählen Sie in der Detailansicht der Umgebungsvorlage im Abschnitt Vorlagenversionen die Option Neue Version erstellen aus.
4. Wählen Sie auf der Seite Neue Version der Umgebungsvorlage erstellen im Abschnitt Quelle des Vorlagenpakets eine der beiden verfügbaren Quelloptionen für das Vorlagenpaket aus.
   - Verwenden Sie eines unserer Beispiel-Template-Bundles.
   - Verwenden Sie Ihr eigenes Vorlagenpaket.
5. Geben Sie einen Pfad zum ausgewählten Vorlagenpaket an.
   - Wenn Sie Eines unserer Mustervorlagenpakete verwenden ausgewählt haben, wählen Sie im Abschnitt Beispielvorlagenpaket ein Mustervorlagenpaket aus.
   - Wenn Sie Ihr eigenes Vorlagenpaket verwenden ausgewählt haben, wählen Sie im Abschnitt Speicherort des S3-Pakets den Pfad zu Ihrem Vorlagenpaket aus.
6. Im Abschnitt Vorlagendetails.
   a. (Optional) Geben Sie einen Anzeigenamen für die Vorlage ein.
   b. (Optional) Geben Sie eine Vorlagenbeschreibung für die Dienstvorlage ein.
7. Wählen Sie im Abschnitt Vorlagendetails eine der folgenden Optionen aus.

- Um eine Nebenversion zu erstellen, lassen Sie das Kontrollkästchen Aktivieren, um eine neue Hauptversion zu erstellen, leer.
- Um eine Hauptversion zu erstellen, aktivieren Sie das Kontrollkästchen Aktivieren, um eine neue Hauptversion zu erstellen.

8. Fahren Sie mit den Konsolenschritten fort, um die neue Neben- oder Hauptversion zu erstellen, und wählen Sie Neue Version erstellen aus.

AWS CLI

Verwenden Sie die CLI, um eine neue Umgebungsvorlage zu registrieren und zu veröffentlichen, wie in den folgenden Schritten gezeigt.

1. Erstellen Sie eine standardmäßige oder vom Kunden verwaltete Umgebungsvorlage, indem Sie die Region, den Namen, den Anzeigenamen (optional) und die Beschreibung (optional) angeben.

   a. Erstellen Sie eine Standardumgebungsvorlage.

      Führen Sie den folgenden Befehl aus:

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

      Antwort:

```
{
    "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. Erstellen Sie eine vom Kunden verwaltete Umgebungsvorlage, indem Sie den `provisioning` Parameter mit Wert hinzufügen`CUSTOMER_MANAGED`.

Führen Sie den folgenden Befehl aus:

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

Antwort:

```
{
    "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. Erstellen Sie eine Nebenversion 0 der Hauptversion 1 der Umgebungsvorlage

Dieser und die übrigen Schritte sind für die standardmäßigen und kundenverwalteten Umgebungsvorlagen identisch.

Geben Sie den Vorlagennamen, die Hauptversion sowie den S3-Bucket-Namen und den Schlüssel für den Bucket an, der Ihr Umgebungsvorlagenpaket enthält.

Führen Sie den folgenden Befehl aus:

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

Antwort:

```
{
    "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. Verwenden Sie den Befehl get, um den Status der Registrierungen zu überprüfen.

   Führen Sie den folgenden Befehl aus:

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

   Antwort:

```
{
    "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. Veröffentlichung der Nebenversion 0 der Hauptversion 1 der Umgebungsvorlage unter Angabe des Vorlagennamens sowie der Haupt- und Nebenversion. Diese Version ist die `Recommended` Version.

   Führen Sie den folgenden Befehl aus:

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

   Antwort:

```
{
    "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"
    }
}
```

Nachdem Sie mit dem eine neue Vorlage erstellt haben AWS CLI, können Sie eine Liste mit AWS und vom Kunden verwalteten Tags anzeigen. AWS Proton generiert automatisch AWS verwaltete Tags für Sie. Mit dem können Sie auch vom Kunden verwaltete Tags ändern und erstellen AWS CLI. Weitere Informationen finden Sie unter AWS Proton Ressourcen und Tagging.

Führen Sie den folgenden Befehl aus:

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

Registrieren und veröffentlichen Sie Servicevorlagen

Wenn Sie eine Service-Vorlagenversion erstellen, geben Sie eine Liste kompatibler Umgebungsvorlagen an. Auf diese Weise haben Entwickler bei der Auswahl einer Dienstvorlage Optionen dafür, in welcher Umgebung sie ihren Dienst bereitstellen möchten.

Stellen Sie vor dem Erstellen eines Dienstes anhand einer Dienstvorlage oder vor dem Veröffentlichen einer Dienstvorlage sicher, dass die Umgebungen anhand der aufgelisteten kompatiblen Umgebungsvorlagen bereitgestellt wurden.

Sie können einen Dienst nicht auf die neue Hauptversion aktualisieren, wenn er in einer Umgebung bereitgestellt wird, die aus einer entfernten kompatiblen Umgebungsvorlage erstellt wurde.

Um kompatible Umgebungsvorlagen für eine Dienstvorlagenversion hinzuzufügen oder zu entfernen, erstellen Sie eine neue Hauptversion davon.

Sie können die Konsole oder die verwenden AWS CLI , um eine Dienstvorlage zu registrieren und zu veröffentlichen.

AWS-Managementkonsole

Verwenden Sie die Konsole, um eine neue Dienstvorlage zu registrieren und zu veröffentlichen.

1. Wählen Sie in der AWS Proton Konsole Service Templates aus.

2. Wählen Sie Dienstvorlage erstellen aus.
3. Wählen Sie auf der Seite „Dienstvorlage erstellen“ im Abschnitt „Quelle des Vorlagenpakets“ eine der verfügbaren Vorlagenoptionen aus.

   - Verwenden Sie Ihr eigenes Vorlagenpaket.
   - Synchronisieren Sie Vorlagen von Git.
4. Geben Sie einen Pfad zu einem Vorlagenpaket an.

   a. Wenn du Vorlagen von Git synchronisieren ausgewählt hast, gehe im Abschnitt Quellcode-Repository wie folgt vor:

      i. Wählen Sie das Repository für Ihre Konfiguration zur Vorlagensynchronisierung aus.

      ii. Geben Sie den Namen des Repository-Zweigs ein, von dem aus synchronisiert werden soll.

      iii. (Optional) Geben Sie den Namen eines Verzeichnisses ein, um die Suche nach Ihrem Vorlagenpaket einzuschränken.

   b. Andernfalls geben Sie im Abschnitt Speicherort des S3-Bundles einen Pfad zu Ihrem Vorlagenpaket an.
5. Im Abschnitt Vorlagendetails.

   a. Geben Sie einen Namen für die Vorlage ein.

   b. (Optional) Geben Sie einen Anzeigenamen für die Vorlage ein.

   c. (Optional) Geben Sie eine Vorlagenbeschreibung für die Dienstvorlage ein.
6. Wählen Sie im Abschnitt Kompatible Umgebungsvorlagen aus einer Liste kompatibler Umgebungsvorlagen aus.
7. (Optional) Wählen Sie im Abschnitt Verschlüsselungseinstellungen die Option Verschlüsselungseinstellungen anpassen (erweitert) aus, um Ihren eigenen Verschlüsselungsschlüssel anzugeben.
8. (Optional) Gehen Sie im Abschnitt Pipeline wie folgt vor:

   Wenn Sie keine Service-Pipeline-Definition in Ihre Service-Vorlage aufnehmen, deaktivieren Sie das Kontrollkästchen Pipeline — optional unten auf der Seite. Sie können dies nicht ändern, nachdem die Dienstvorlage erstellt wurde. Weitere Informationen finden Sie unter Vorlagenpakete.

9. (Optional) Wählen Sie im Abschnitt Unterstützte Komponentenquellen für Komponentenquellen die Option Direkt definiert aus, um das Anhängen direkt definierter Komponenten an Ihre Dienstinstanzen zu ermöglichen.
10. (Optional) Wählen Sie im Abschnitt „Tags“ die Option Neues Tag hinzufügen aus und geben Sie einen Schlüssel und einen Wert ein, um ein vom Kunden verwaltetes Tag zu erstellen.
11. Wählen Sie „Servicevorlage erstellen“ aus.

    Sie befinden sich jetzt auf einer neuen Seite, auf der der Status und die Details für Ihre neue Servicevorlage angezeigt werden. Zu diesen Details gehören eine Liste von AWS und vom Kunden verwaltete Tags. AWS Proton generiert automatisch AWS verwaltete Tags für Sie, wenn Sie AWS Proton Ressourcen erstellen. Weitere Informationen finden Sie unter AWS Proton Ressourcen und Tagging.
12. Der Status einer neuen Dienstvorlage beginnt im Status Entwurf. Sie und andere Personen mit `proton:CreateService` entsprechenden Berechtigungen können die Datei anzeigen und darauf zugreifen. Folgen Sie dem nächsten Schritt, um die Vorlage für andere verfügbar zu machen.
13. Wählen Sie im Abschnitt Vorlagenversionen das Optionsfeld links neben der Nebenversion der Vorlage, die Sie gerade erstellt haben (1.0). Alternativ können Sie in der Informationswarnung die Option Veröffentlichen auswählen und den nächsten Schritt überspringen.
14. Wählen Sie im Abschnitt Vorlagenversionen die Option Veröffentlichen aus.
15. Der Status der Vorlage ändert sich in „Veröffentlicht“. Da es sich um die neueste Version der Vorlage handelt, handelt es sich um die empfohlene Version.
16. Wählen Sie im Navigationsbereich Dienstvorlagen aus, um eine Liste Ihrer Dienstvorlagen und Details anzuzeigen.

Verwenden Sie die Konsole, um neue Haupt- und Nebenversionen einer Dienstvorlage zu registrieren.

Weitere Informationen finden Sie unter Versionierte Vorlagen.

1. Wählen Sie in der AWS Proton Konsole Service Templates aus.
2. Wählen Sie in der Liste der Dienstvorlagen den Namen der Dienstvorlage aus, für die Sie eine Haupt- oder Nebenversion erstellen möchten.

3. Wählen Sie in der Detailansicht der Servicevorlage im Abschnitt Vorlagenversionen die Option Neue Version erstellen aus.
4. Wählen Sie auf der Seite Neue Service-Vorlagenversion erstellen im Abschnitt Bundle-Quelle die Option Eigenes Vorlagenpaket verwenden aus.
5. Wählen Sie im Abschnitt Speicherort des S3-Bundles den Pfad zu Ihrem Vorlagenpaket aus.
6. Im Abschnitt Vorlagendetails.
   a. (Optional) Geben Sie einen Anzeigenamen für die Vorlage ein.
   b. (Optional) Geben Sie eine Vorlagenbeschreibung für die Dienstvorlage ein.
7. Wählen Sie im Abschnitt Vorlagendetails eine der folgenden Optionen aus.
   - Um eine Nebenversion zu erstellen, lassen Sie das Kontrollkästchen Aktivieren, um eine neue Hauptversion zu erstellen, leer.
   - Um eine Hauptversion zu erstellen, aktivieren Sie das Kontrollkästchen Aktivieren, um eine neue Hauptversion zu erstellen.
8. Fahren Sie mit den Konsolenschritten fort, um die neue Neben- oder Hauptversion zu erstellen, und wählen Sie Neue Version erstellen aus.

AWS CLI

Um eine Dienstvorlage zu erstellen, die einen Service ohne Service-Pipeline bereitstellt, fügen Sie dem `create-service-template` Befehl den Parameter und den Wert `--pipeline-provisioning "CUSTOMER_MANAGED"` hinzu. Konfigurieren Sie Ihre Vorlagenpakete wie unter Vorlagenpakete Erstellung und beschrieben. Schemaanforderungen für Servicevorlagenpakete

> **Note**
>
> `pipelineProvisioning`Nachdem die Dienstvorlage erstellt wurde, können Sie sie nicht mehr ändern.

1. Verwenden Sie die CLI, um eine neue Dienstvorlage mit oder ohne Service-Pipeline zu registrieren und zu veröffentlichen, wie in den folgenden Schritten gezeigt.

   a. Erstellen Sie mithilfe der CLI eine Dienstvorlage mit einer Service-Pipeline.

      Geben Sie den Namen, den Anzeigenamen (optional) und die Beschreibung (optional) an.

      Führen Sie den folgenden Befehl aus:

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

      Antwort:

```
{
    "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. Erstellen Sie eine Dienstvorlage ohne Dienstpipeline.

      Fügen Sie `--pipeline-provisioning` hinzu.

      Führen Sie den folgenden Befehl aus:

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

      Antwort:

```
{
    "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. Erstellen Sie eine Nebenversion 0 der Hauptversion 1 der Dienstvorlage.

   Geben Sie den Vorlagennamen, die kompatiblen Umgebungsvorlagen, die Hauptversion sowie den S3-Bucket-Namen und -Schlüssel für den Bucket an, der Ihr Service-Vorlagenpaket enthält.

   Führen Sie den folgenden Befehl aus:

```
$ 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"}]'
```

   Antwort:

```
{
    "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. Verwenden Sie den Befehl get, um den Status der Registrierungen zu überprüfen.

   Führen Sie den folgenden Befehl aus:

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

   Antwort:

```
{
    "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. Veröffentlichen Sie die Dienstvorlage, indem Sie den Befehl update verwenden, um den Status zu zu "PUBLISHED" ändern.

   Führen Sie den folgenden Befehl aus:

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

   Antwort:

```
{
    "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. Überprüfen Sie, ob Version 1.0 veröffentlicht AWS Proton wurde, indem Sie den Befehl get verwenden, um die Detaildaten der Service-Vorlage abzurufen.

   Führen Sie den folgenden Befehl aus:

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

   Antwort:

```
{
    "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"
    }
}
```


Vorlagendaten anzeigen

Mithilfe der AWS Proton Konsole und können Sie Vorlagenlisten mit Details und einzelne Vorlagen mit Detaildaten anzeigen AWS CLI.

Zu den Vorlagendaten für vom Kunden verwaltete Umgebungen gehört der `provisioned` Parameter mit dem Wert`CUSTOMER_MANAGED`.

Wenn eine Dienstvorlage keine Service-Pipeline enthält, enthalten die Service-Vorlagendaten den `pipelineProvisioning` Parameter mit dem Wert`CUSTOMER_MANAGED`.

Weitere Informationen finden Sie unter Vorlagen registrieren und veröffentlichen.

Sie können die Konsole oder die verwenden AWS CLI , um Vorlagendaten aufzulisten und anzuzeigen.

AWS-Managementkonsole

Verwenden Sie die Konsole, um Vorlagen aufzulisten und anzuzeigen.

1. Um eine Liste mit Vorlagen anzuzeigen, wählen Sie Vorlagen (Umgebung oder Dienst).
2. Um Detaildaten anzuzeigen, wählen Sie den Namen einer Vorlage.

   Zeigen Sie die Detaildaten der Vorlage, eine Liste der Haupt- und Nebenversionen der Vorlage sowie eine Liste der AWS Proton Ressourcen an, die mithilfe von Vorlagenversionen und Vorlagen-Tags bereitgestellt wurden.

   Die empfohlene Haupt- und Nebenversion ist als Empfohlen gekennzeichnet.

AWS CLI

Verwenden Sie die AWS CLI , um Vorlagen aufzulisten und anzuzeigen.

Führen Sie den folgenden Befehl aus:

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

Antwort:

```
{
    "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"
    }
}
```

Führen Sie den folgenden Befehl aus:

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

Antwort:

```
{
    "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"
        }
    ]
}
```

Eine Nebenversion einer Dienstvorlage anzeigen.

Führen Sie den folgenden Befehl aus:

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

Antwort:


```
{
    "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"
    }
}
```

Zeigen Sie eine Dienstvorlage ohne Service-Pipeline an, wie im nächsten Beispiel für Befehl und Antwort gezeigt.

Führen Sie den folgenden Befehl aus:

```
$ aws proton get-service-template \
    --name "simple-svc-template-cli"
```

Antwort:

```
{
    "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"
    }
}
```

Aktualisieren Sie eine Vorlage

Sie können eine Vorlage wie in der folgenden Liste beschrieben aktualisieren.

- Bearbeiten Sie das `description` Oder `display name` einer Vorlage, wenn Sie entweder die Konsole oder verwenden AWS CLI. Sie können das *name* einer Vorlage nicht bearbeiten.
- Aktualisieren Sie den Status einer Nebenversion einer Vorlage, wenn Sie entweder die Konsole oder verwenden AWS CLI. Sie können den Status nur von `DRAFT` bis ändern`PUBLISHED`.
- Bearbeiten Sie den Anzeigenamen und die Beschreibung einer Neben- oder Hauptversion einer Vorlage, wenn Sie die verwenden AWS CLI.

AWS-Managementkonsole

Bearbeiten Sie eine Vorlagenbeschreibung und einen Anzeigenamen mithilfe der Konsole, wie in den folgenden Schritten beschrieben.

In der Liste der Vorlagen.

1. Wählen Sie in der AWS Proton Konsole die Option (Umgebung oder Dienst) Templates aus.
2. Wählen Sie in der Liste der Vorlagen das Optionsfeld links neben der Vorlage aus, für die Sie die Beschreibung oder den Anzeigenamen aktualisieren möchten.
3. Wählen Sie „Aktionen“ und dann „Bearbeiten“.
4. Geben Sie auf der Seite Vorlage bearbeiten (Umgebung oder Service) im Abschnitt Vorlagendetails Ihre Änderungen in das Formular ein und wählen Sie Änderungen speichern aus.

Ändern Sie den Status einer Nebenversion einer Vorlage mithilfe der Konsole, um eine Vorlage zu veröffentlichen, wie im Folgenden beschrieben. Sie können den Status nur von `DRAFT` bis ändern`PUBLISHED`.

Auf der Detailseite der Vorlage (Umgebung oder Service).

1. Wählen Sie in der AWS Proton Konsole die Vorlagen (Umgebung oder Dienst) aus.
2. Wählen Sie in der Liste der Vorlagen den Namen der Vorlage aus, für die Sie den Status einer Nebenversion von Entwurf auf Veröffentlicht aktualisieren möchten.
3. Wählen Sie auf der Detailseite der Vorlage (Umgebung oder Service) im Abschnitt Vorlagenversionen das Optionsfeld links neben der Nebenversion aus, die Sie veröffentlichen möchten.
4. Wählen Sie im Abschnitt Vorlagenversionen die Option Veröffentlichen aus. Der Status ändert sich von Entwurf zu Veröffentlicht.

AWS CLI

Der folgende Beispielbefehl und die folgende Antwort zeigen, wie Sie die Beschreibung einer Umgebungsvorlage bearbeiten können.

Führen Sie den folgenden Befehl aus.

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

Antwort:

```
{
    "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"
    }
}
```

Sie können die auch verwenden AWS CLI , um Dienstvorlagen zu aktualisieren. Registrieren und veröffentlichen Sie ServicevorlagenIn Schritt 5 finden Sie ein Beispiel für die Aktualisierung des Status einer Nebenversion einer Dienstvorlage.

Vorlagen löschen

Vorlagen können mit der Konsole und gelöscht werden AWS CLI.

Sie können eine Nebenversion einer Umgebungsvorlage löschen, wenn für diese Version keine Umgebungen bereitgestellt wurden.

Sie können eine Nebenversion einer Dienstvorlage löschen, wenn für diese Version keine Dienstinstanzen oder Pipelines bereitgestellt wurden. Ihre Pipeline kann in einer anderen Vorlagenversion als Ihre Serviceinstanz bereitgestellt werden. Wenn Ihre Serviceinstanz beispielsweise von 1.0 auf Version 1.1 aktualisiert wurde und Ihre Pipeline immer noch auf Version 1.0 bereitgestellt wird, können Sie die Dienstvorlage 1.0 nicht löschen.

AWS-Managementkonsole

Sie können die Konsole verwenden, um die gesamte Vorlage oder einzelne Neben- und Hauptversionen einer Vorlage zu löschen.

Verwenden Sie die Konsole, um Vorlagen wie folgt zu löschen.

> **Note**
>
> Wenn Sie die Konsole zum Löschen von Vorlagen verwenden.
>
> - Wenn Sie die gesamte Vorlage löschen, löschen Sie auch die Haupt- und Nebenversionen der Vorlage.

In der Liste der (Umgebungs- oder Service-) Vorlagen.

1. Wählen Sie in der AWS Proton Konsole die Option (Umgebungs- oder Service-) Vorlagen aus.
2. Wählen Sie in der Liste der Vorlagen das Optionsfeld links neben der Vorlage aus, die Sie löschen möchten.

   Sie können eine gesamte Vorlage nur löschen, wenn für ihre Versionen keine AWS Proton Ressourcen bereitgestellt wurden.
3. Wählen Sie Aktionen und anschließend Löschen, um die gesamte Vorlage zu löschen.
4. In einem Modal werden Sie aufgefordert, die Löschaktion zu bestätigen.
5. Folgen Sie den Anweisungen und wählen Sie Ja, löschen.

Auf der Detailseite der Vorlage (Umgebung oder Service).

1. Wählen Sie in der AWS Proton Konsole Vorlagen für (Umgebung oder Dienst) aus.
2. Wählen Sie in der Liste der Vorlagen den Namen der Vorlage aus, die Sie vollständig löschen möchten, oder löschen Sie einzelne Haupt- oder Nebenversionen davon.
3. Um die gesamte Vorlage zu löschen.

   Sie können eine gesamte Vorlage nur löschen, wenn für ihre Versionen keine AWS Proton Ressourcen bereitgestellt wurden.

a. Wählen Sie in der oberen rechten Ecke der Seite Löschen.
b. In einem Modal werden Sie aufgefordert, die Löschaktion zu bestätigen.
c. Folgen Sie den Anweisungen und wählen Sie Ja, löschen.

4. Um Haupt- oder Nebenversionen einer Vorlage zu löschen.

   Sie können eine Nebenversion einer Vorlage nur löschen, wenn für diese Version keine AWS Proton Ressourcen bereitgestellt wurden.

   a. Wählen Sie im Abschnitt Vorlagenversionen das Optionsfeld links neben der Version aus, die Sie löschen möchten.
   b. Wählen Sie im Abschnitt Vorlagenversionen die Option Löschen aus.
   c. In einem Modal werden Sie aufgefordert, die Löschaktion zu bestätigen.
   d. Folgen Sie den Anweisungen und wählen Sie Ja, löschen.

AWS CLI

AWS CLI Vorgänge zum Löschen von Vorlagen beinhalten nicht das Löschen anderer Versionen einer Vorlage. Wenn Sie die verwenden AWS CLI, löschen Sie Vorlagen unter den folgenden Bedingungen.

- Löschen Sie eine gesamte Vorlage, wenn keine Neben- oder Hauptversionen der Vorlage vorhanden sind.
- Löschen Sie eine Hauptversion, wenn Sie die letzte verbleibende Nebenversion löschen.
- Löschen Sie eine Nebenversion einer Vorlage, wenn für diese Version keine AWS Proton Ressourcen bereitgestellt wurden.
- Löschen Sie die empfohlene Nebenversion einer Vorlage, wenn keine anderen Nebenversionen der Vorlage vorhanden sind und für diese Version keine AWS Proton Ressourcen bereitgestellt wurden.

Die folgenden Beispielbefehle und Antworten zeigen, wie Sie Vorlagen mithilfe von löschen AWS CLI können.

Führen Sie den folgenden Befehl aus:

```
$ aws proton delete-environment-template-version \
    --template-name "simple-env" \
```

```
    --major-version "1" \
    --minor-version "0"
```

Antwort:

```
{
    "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"
    }
}
```

Führen Sie den folgenden Befehl aus:

```
$ aws proton delete-environment-template \
    --name "simple-env"
```

Antwort:

```
{
    "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"
    }
}
```

Führen Sie den folgenden Befehl aus:

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

Antwort:

```
{
    "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"
    }
}
```

Konfigurationen für die Vorlagensynchronisierung

Erfahre, wie du ein Template so konfigurierst, dass es aus Template-Bundles AWS Proton synchronisiert werden kann, die sich in registrierten Git-Repositorys befinden, die du definierst. Wenn ein Commit in dein Repository gepusht wird, AWS Proton überprüft es, ob Änderungen an deinen Repository-Vorlagenpaketen vorgenommen wurden. Wenn es eine Änderung des Template-Bundles feststellt, wird eine neue Neben- oder Hauptversion seines Templates erstellt, sofern die Version noch nicht existiert. AWS Proton unterstützt GitHub derzeit GitHub Enterprise und BitBucket.

Einen Commit an ein synchronisiertes Template-Bundle weiterleiten

Wenn du einen Commit in einen Branch überträgst, der von einer deiner Vorlagen verfolgt wird, wird dein Repository AWS Proton geklont und bestimmt, welche Vorlagen synchronisiert werden müssen. Es durchsucht die Dateien in deinem Verzeichnis nach Verzeichnissen, die der Konvention von `{template-name}/{major-version}/` entsprechen.

Nachdem AWS Proton festgestellt wurde, welche Vorlagen und Hauptversionen mit Ihrem Repository und Branch verknüpft sind, versucht es, all diese Vorlagen parallel zu synchronisieren.

Bei jeder Synchronisierung mit einer bestimmten Vorlage wird AWS Proton zunächst geprüft, ob sich der Inhalt des Vorlagenverzeichnisses seit der letzten erfolgreichen Synchronisierung geändert hat. Wenn sich der Inhalt nicht geändert hat, wird die Registrierung eines doppelten Bundles AWS Proton übersprungen. Dadurch wird sichergestellt, dass eine neue Nebenversion der Vorlage erstellt wird, wenn sich der Inhalt des Vorlagenpakets ändert. Wenn sich der Inhalt des Vorlagenpakets geändert hat, wird das Bundle bei registriert AWS Proton.

AWS Proton Überwacht nach der Registrierung des Vorlagenpakets den Registrierungsstatus, bis die Registrierung abgeschlossen ist.

Es kann jeweils nur eine Synchronisierung mit einer bestimmten Neben- und Hauptversion der Vorlage erfolgen. Alle Commits, die während einer laufenden Synchronisierung möglicherweise per Push übertragen wurden, werden gebündelt. Die gebündelten Commits werden synchronisiert, nachdem der vorherige Synchronisierungsversuch abgeschlossen ist.

Dienstvorlagen werden synchronisiert

AWS Proton kann sowohl Umgebungs- als auch Dienstvorlagen aus Ihrem Git-Repository synchronisieren. Um Ihre Service-Vorlagen zu synchronisieren, fügen Sie jedem Hauptversionsverzeichnis in Ihrem Vorlagenpaket eine zusätzliche Datei mit dem Namen `.template-registration.yaml` hinzu. Diese Datei enthält zusätzliche Informationen, die Sie AWS Proton benötigen, wenn sie nach einem Commit eine Service-Vorlagenversion für Sie erstellt: kompatible Umgebungen und unterstützte Komponentenquellen.

Der vollständige Pfad der Datei lautet*`service-template-name`*/*`major-version`*/`.template-registration.yaml`. Weitere Informationen finden Sie unter the section called “Synchronisieren von Dienstvorlagen”.

Überlegungen zur Konfiguration der Vorlagensynchronisierung

Lesen Sie die folgenden Überlegungen zur Verwendung von Vorlagensynchronisierungskonfigurationen.

- Repositorys dürfen nicht größer als 250 MB sein.
- Um die Vorlagensynchronisierung zu konfigurieren, verknüpfen Sie zunächst das Repository mit AWS Proton. Weitere Informationen finden Sie unter the section called “Einen Repository-Link erstellen”.

- Wenn aus einer synchronisierten Vorlage eine neue Vorlagenversion erstellt wird, befindet sie sich im DRAFT Status.
- Eine neue Nebenversion einer Vorlage wird erstellt, wenn eine der folgenden Bedingungen zutrifft:
  - Der Inhalt des Vorlagenpakets unterscheidet sich von denen der letzten synchronisierten Vorlagennebenversion.
  - Die letzte zuvor synchronisierte Nebenversion der Vorlage wurde gelöscht.
- Die Synchronisierung kann nicht angehalten werden.
- Sowohl neue Neben- als auch Hauptversionen werden automatisch synchronisiert.
- Neue Vorlagen der obersten Ebene können nicht durch Konfigurationen zur Vorlagensynchronisierung erstellt werden.
- Mit einer Konfiguration zur Vorlagensynchronisierung können Sie nicht mit einer Vorlage aus mehreren Repositorys synchronisieren.
- Sie können keine Tags anstelle von Branches verwenden.
- Wenn Sie eine Dienstvorlage erstellen, geben Sie kompatible Umgebungsvorlagen an.
- Sie können eine Umgebungsvorlage erstellen und sie als kompatible Umgebung für Ihre Dienstvorlage im selben Commit hinzufügen.
- Synchronisierungen mit einer einzigen Hauptversion der Vorlage werden nacheinander ausgeführt. Wenn während einer Synchronisierung neue Commits erkannt werden, werden sie gebündelt und am Ende der aktiven Synchronisierung angewendet. Synchronisierungen mit verschiedenen Hauptversionen von Vorlagen erfolgen parallel.
- Wenn Sie den Zweig ändern, aus dem Ihre Vorlagen synchronisiert werden, werden alle laufenden Synchronisierungen aus dem alten Zweig zuerst abgeschlossen. Dann beginnt die Synchronisierung mit dem neuen Zweig.
- Wenn Sie das Repository ändern, von dem aus Ihre Vorlagen synchronisiert werden, schlagen alle laufenden Synchronisierungen aus dem alten Repository möglicherweise fehl oder werden vollständig ausgeführt. Das hängt davon ab, in welcher Phase der Synchronisierung sie sich befinden.

Weitere Informationen finden Sie in der Referenz zur AWS Proton Service-API.

Topics

- Bearbeiten Sie eine Konfiguration für die Vorlagensynchronisierung
- Löschen Sie eine Vorlagensynchronisierungskonfiguration

Erstellen Sie eine Konfiguration für die Vorlagensynchronisierung

Erfahren Sie, wie Sie eine Vorlagen-Synchronisierungskonfiguration mit erstellen AWS Proton.

Voraussetzungen für die Synchronisation einer Vorlage erstellen:

- Sie haben ein Repository mit verknüpft AWS Proton.
- Ein Vorlagenpaket befindet sich in Ihrem Repository.

Der Repository-Link besteht aus folgenden Elementen:

- Eine CodeConnections Verbindung, die dir AWS Proton erlaubt, auf dein Repository zuzugreifen und dessen Benachrichtigungen zu abonnieren.
- Eine mit einem Dienst verknüpfte Rolle. Wenn Sie Ihr Repository verknüpfen, wird die mit dem Service verknüpfte Rolle für Sie erstellt.

Bevor Sie Ihre erste Konfiguration für die Vorlagensynchronisierung erstellen, übertragen Sie ein Vorlagenpaket in Ihr Repository, wie im folgenden Verzeichnislayout dargestellt.

```
/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/
```

Nachdem Sie Ihre erste Konfiguration für die Vorlagensynchronisierung erstellt haben, werden neue Vorlagenversionen automatisch erstellt, wenn Sie einen Commit pushen, der ein aktualisiertes Vorlagenpaket unter einer neuen Version hinzufügt (z. B. unter`/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/
```

Sie können neue Template-Bundle-Versionen für eine oder mehrere synchronisierte Vorlagen in einen einzigen Commit aufnehmen. AWS Proton erstellt eine neue Vorlagenversion für jede neue Version des Vorlagenpakets, die im Commit enthalten war.

Nachdem Sie die Konfiguration für die Vorlagensynchronisierung erstellt haben, können Sie immer noch manuell neue Versionen der Vorlage in der Konsole oder mit der erstellen, AWS CLI indem Sie Vorlagenpakete aus einem S3-Bucket hochladen. Die Vorlagensynchronisierung funktioniert nur in eine Richtung: von Ihrem Repository zu. AWS Proton Manuell erstellte Vorlagenversionen werden nicht synchronisiert.

Nachdem Sie eine Konfiguration für die Vorlagensynchronisierung eingerichtet haben AWS Proton , wartet auf Änderungen an Ihrem Repository. Immer wenn eine Änderung übertragen wird, wird nach einem Verzeichnis gesucht, das denselben Namen wie Ihre Vorlage hat. Es sucht dann in diesem Verzeichnis nach Verzeichnissen, die wie Hauptversionen aussehen. AWS Proton registriert das Vorlagenpaket für die entsprechende Hauptversion der Vorlage. Die neuen Versionen befinden sich immer im `DRAFT` Status. Sie können die neuen Versionen mit der Konsole oder veröffentlichen AWS CLI.

Nehmen wir zum Beispiel an, Sie haben eine Vorlage, die aufgerufen wird und `my-env-template` so konfiguriert ist, dass sie von `main` einem Zweig aus `my-repo/templates` synchronisiert wird, mit dem folgenden Layout.

```
/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 synchronisiert den Inhalt von `/templates/my-env-template/v1/` to `my-env-template:1` und den Inhalt von `/templates/my-env-template/v2/` tomy`-env-template:2`. Falls sie noch nicht existieren, werden diese Hauptversionen erstellt.

AWS Proton hat das erste Verzeichnis gefunden, das dem Vorlagennamen entsprach. Sie können die AWS Proton Suche nach Verzeichnissen einschränken, indem Sie `subdirectoryPath` beim Erstellen oder Bearbeiten einer Vorlagensynchronisierungskonfiguration eine angeben. Sie können beispielsweise `/production-templates/` für angeben`subdirectoryPath`.

Sie können mit der Konsole oder der CLI eine Vorlagensynchronisierungskonfiguration erstellen.

AWS-Managementkonsole

Erstellen Sie mithilfe der Konsole eine Vorlage und eine Konfiguration für die Vorlagensynchronisierung.

1. Wählen Sie in der AWS Proton Konsole die Option Umgebungsvorlagen aus.
2. Wählen Sie Umgebungsvorlage erstellen aus.
3. Wählen Sie auf der Seite Umgebungsvorlage erstellen im Abschnitt Vorlagenoptionen die Option Vorlage für die Bereitstellung neuer Umgebungen erstellen aus.
4. Wählen Sie im Quellbereich des Vorlagenpakets die Option Vorlagen aus Git synchronisieren aus.
5. Gehen Sie im Abschnitt Quellcode-Repository wie folgt vor:
   a. Wählen Sie für Repository das verknüpfte Repository aus, das Ihr Vorlagenpaket enthält.
   b. Wählen Sie für Branch einen Repository-Branch aus, von dem aus synchronisiert werden soll.
   c. (Optional) Geben Sie für das Verzeichnis der Vorlagenpakete den Namen eines Verzeichnisses ein, um die Suche nach Ihrem Vorlagenpaket einzugrenzen.
6. Im Abschnitt Vorlagendetails.
   a. Geben Sie einen Namen für die Vorlage ein.
   b. (Optional) Geben Sie einen Anzeigenamen für die Vorlage ein.
   c. (Optional) Geben Sie eine Vorlagenbeschreibung für die Umgebungsvorlage ein.
7. (Optional) Aktivieren Sie im Abschnitt Verschlüsselungseinstellungen das Kontrollkästchen Verschlüsselungseinstellungen anpassen (erweitert), um Ihren eigenen Verschlüsselungsschlüssel anzugeben.
8. (Optional) Wählen Sie im Abschnitt Tags die Option Neues Tag hinzufügen aus und geben Sie einen Schlüssel und einen Wert ein, um ein vom Kunden verwaltetes Tag zu erstellen.
9. Wählen Sie „Umgebungsvorlage erstellen“.

Sie befinden sich jetzt auf einer neuen Seite, auf der der Status und die Details für Ihre neue Umgebungsvorlage angezeigt werden. Zu diesen Details gehört eine Liste der AWS verwalteten und vom Kunden verwalteten Tags. AWS Proton generiert automatisch AWS verwaltete Tags für Sie, wenn Sie AWS Proton Ressourcen erstellen. Weitere Informationen finden Sie unter AWS Proton Ressourcen und Tagging.

10. Wählen Sie auf der Seite mit den Vorlagendetails die Registerkarte Synchronisieren aus, um Detaildaten zur Konfiguration der Vorlagensynchronisierung anzuzeigen.
11. Wählen Sie den Tab Vorlagenversionen, um Vorlagenversionen mit Statusdetails anzuzeigen.
12. Der Status einer neuen Umgebungsvorlage beginnt im Status Entwurf. Sie und andere Personen mit `proton:CreateEnvironment` entsprechenden Berechtigungen können sie anzeigen und darauf zugreifen. Folgen Sie dem nächsten Schritt, um die Vorlage für andere verfügbar zu machen.
13. Wählen Sie im Abschnitt Vorlagenversionen das Optionsfeld links neben der Nebenversion der Vorlage, die Sie gerade erstellt haben (1.0). Alternativ können Sie in der Informationswarnung die Option Veröffentlichen auswählen und den nächsten Schritt überspringen.
14. Wählen Sie im Abschnitt Vorlagenversionen die Option Veröffentlichen aus.
15. Der Status der Vorlage ändert sich in Veröffentlicht. Es ist die neueste und empfohlene Version der Vorlage.
16. Wählen Sie im Navigationsbereich Umgebungsvorlagen aus, um eine Liste Ihrer Umgebungsvorlagen und Details anzuzeigen.

Das Verfahren zum Erstellen einer Dienstvorlage und einer Konfiguration für die Vorlagensynchronisierung ist ähnlich.

AWS CLI

Erstellen Sie eine Vorlage und eine Konfiguration für die Vorlagensynchronisierung mithilfe von AWS CLI.

1. Erstellen Sie eine Vorlage. In diesem Beispiel wird eine Umgebungsvorlage erstellt.

   Führen Sie den folgenden Befehl aus.

```
$ aws proton create-environment-template \
    --name "env-template"
```

Die Antwort lautet wie folgt.

```
{
    "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. Erstellen Sie Ihre Vorlagen-Synchronisierungskonfiguration mit, AWS CLI indem Sie Folgendes angeben:

   - Die Vorlage, mit der Sie synchronisieren möchten. Nachdem Sie die Konfiguration für die Vorlagensynchronisierung erstellt haben, können Sie daraus immer noch manuell in der Konsole oder mit dem neue Versionen erstellen AWS CLI.
   - Der Name der Vorlage.
   - Der Vorlagentyp.
   - Das verknüpfte Repository, aus dem Sie synchronisieren möchten.
   - Der Anbieter des verknüpften Repositorys.
   - Der Zweig, in dem sich das Vorlagenpaket befindet.
   - (Optional) Der Pfad zu dem Verzeichnis, das Ihr Vorlagenpaket enthält. Sucht standardmäßig nach dem ersten Verzeichnis, das Ihrem Vorlagennamen entspricht. AWS Proton

   Führen Sie den folgenden Befehl aus.

```
$ 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/"
```

Die Antwort lautet wie folgt.

```
{
    "templateSyncConfigDetails": {
        "branch": "main",
        "repositoryName": "myrepos/templates",
        "repositoryProvider": "GITHUB",
        "subdirectory": "templates",
        "templateName": "env-template",
        "templateType": "ENVIRONMENT"
    }
}
```

3. Informationen zum Veröffentlichen Ihrer Vorlagenversion finden Sie unterVorlagen registrieren und veröffentlichen.

Synchronisieren von Dienstvorlagen

Die vorherigen Beispiele zeigen, wie Sie Umgebungsvorlagen synchronisieren können. Dienstvorlagen sind ähnlich. Um Dienstvorlagen zu synchronisieren, fügen Sie jedem Hauptversionsverzeichnis in Ihrem Vorlagenpaket eine zusätzliche Datei mit dem Namen `.template-registration.yaml` hinzu. Diese Datei enthält zusätzliche Details, die AWS Proton benötigt werden, wenn sie nach einem Commit eine Service-Vorlagenversion für Sie erstellt. Wenn Sie mithilfe der AWS Proton Konsole oder API explizit eine Service-Vorlagenversion erstellen, geben Sie diese Details als Eingaben an, und diese Datei ersetzt diese Eingaben für die Vorlagensynchronisierung.

```
./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/
```

Die `.template-registration.yaml` Datei enthält die folgenden Details:

- Kompatible Umgebungen [erforderlich] — Umgebungen, die auf diesen Umgebungsvorlagen und Hauptversionen basieren, sind mit Diensten kompatibel, die auf dieser Dienstvorlagenversion basieren.
- Unterstützte Komponentenquellen [optional] — Komponenten, die diese Quellen verwenden, sind mit Diensten kompatibel, die auf dieser Dienstvorlagenversion basieren. Wenn nicht angegeben, können Komponenten nicht an diese Dienste angehängt werden. Weitere Informationen zu Komponenten finden Sie unterKomponenten.

Die YAML-Syntax der Datei lautet wie folgt:

```
compatible_environments:
  - env-templ-name:major-version
  - ...
supported_component_sources:
  - DIRECTLY_DEFINED
```

Geben Sie eine oder mehrere Kombinationen aus Umgebungsvorlage und Hauptversion an. Die Angabe `supported_component_sources` ist optional und der einzige unterstützte Wert ist`DIRECTLY_DEFINED`.

Example.template-registration.yaml

In diesem Beispiel ist die Version der Dienstvorlage mit den Hauptversionen 1 und 2 der Umgebungsvorlage kompatibel. `my-env-template` Sie ist auch mit den Hauptversionen 1 und 3 der `another-env-template` Umgebungsvorlage kompatibel. In der Datei ist nichts angegeben`supported_component_sources`, sodass Komponenten nicht an Dienste angehängt werden können, die auf dieser Version der Dienstvorlage basieren.

```
compatible_environments:
  - my-env-template:1
  - my-env-template:2
  - another-env-template:1
  - another-env-template:3
```

> **Note**
>
> Zuvor wurde eine andere Datei zur Angabe kompatibler Umgebungen AWS Proton definiert. `.compatible-envs` AWS Proton unterstützt diese Datei und ihr Format aus Gründen der

Abwärtskompatibilität weiterhin. Wir empfehlen, es nicht mehr zu verwenden, da es nicht erweiterbar ist und neuere Funktionen wie Komponenten nicht unterstützt.

Konfigurationsdetails für die Vorlagensynchronisierung anzeigen

Zeigen Sie die Konfigurationsdetails der Vorlagensynchronisierung mithilfe der Konsole oder der CLI an.

AWS-Managementkonsole

Verwenden Sie die Konsole, um die Konfigurationsdetails der Vorlagensynchronisierung anzuzeigen.

1. Wählen Sie im Navigationsbereich Vorlagen (Umgebung oder Dienst) aus.
2. Um Detaildaten anzuzeigen, wählen Sie den Namen einer Vorlage aus, für die Sie eine Vorlagensynchronisierungskonfiguration erstellt haben.
3. Wählen Sie auf der Detailseite der Vorlage die Registerkarte Synchronisieren aus, um die Detaildaten der Template-Sync-Konfiguration anzuzeigen.

AWS CLI

Verwenden Sie den AWS CLI , um eine synchronisierte Vorlage anzuzeigen.

Führen Sie den folgenden Befehl aus.

```
$ aws proton get-template-sync-config \
    --template-name "svc-template" \
    --template-type "SERVICE"
```

Die Antwort lautet wie folgt.

```
{
    "templateSyncConfigDetails": {
        "branch": "main",
        "repositoryProvider": "GITHUB",
        "repositoryName": "myrepos/myrepo",
        "subdirectory": "svc-template",
        "templateName": "svc-template",
```

```
        "templateType": "SERVICE"
    }
}
```

Verwenden Sie den AWS CLI , um den Status der Vorlagensynchronisierung abzurufen.

Geben Sie für `template-version` die Hauptversion der Vorlage ein.

Führen Sie den folgenden Befehl aus.

```
$ aws proton get-template-sync-status \
    --template-name "env-template" \
    --template-type "ENVIRONMENT" \
    --template-version "1"
```

Bearbeiten Sie eine Konfiguration für die Vorlagensynchronisierung

Sie können alle Konfigurationsparameter für die Vorlagensynchronisierung mit Ausnahme von `template-name` und bearbeiten`template-type`.

Erfahren Sie, wie Sie eine Vorlagensynchronisierungskonfiguration mithilfe der Konsole oder der CLI bearbeiten.

AWS-Managementkonsole

Bearbeiten Sie einen Konfigurationszweig für die Vorlagensynchronisierung mithilfe der Konsole.

In der Liste der Vorlagen.

1. Wählen Sie in der AWS Proton Konsole Vorlagen (Umgebung oder Dienst) aus.
2. Wählen Sie in der Liste der Vorlagen den Namen der Vorlage mit der Vorlagensynchronisierungskonfiguration aus, die Sie bearbeiten möchten.
3. Wählen Sie auf der Vorlagendetailseite den Tab Vorlagensynchronisierung aus.
4. Wählen Sie im Abschnitt Details zur Vorlagensynchronisierung die Option Bearbeiten aus.
5. Wählen Sie auf der Seite Bearbeiten im Abschnitt Quellcode-Repository für Branch einen Branch aus und klicken Sie dann auf Konfiguration speichern.

AWS CLI

Der folgende Beispielbefehl und die folgende Antwort zeigen, wie Sie eine Vorlagensynchronisierungskonfiguration **`branch`** mit der CLI bearbeiten können.

Führen Sie den folgenden Befehl aus.

```
$ 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"
```

Die Antwort lautet wie folgt.

```
{
    "templateSyncConfigDetails": {
        "branch": "fargate",
        "repositoryProvider": "GITHUB",
        "repositoryName": "myrepos/myrepo",
        "subdirectory": "templates",
        "templateName": "env-template",
        "templateType": "ENVIRONMENT"
    }
}
```

Auf ähnliche Weise können Sie die verwenden AWS CLI , um synchronisierte Dienstvorlagen zu aktualisieren.

Löschen Sie eine Vorlagensynchronisierungskonfiguration

Löschen Sie eine Vorlagensynchronisierungskonfiguration mithilfe der Konsole oder CLI.

AWS-Managementkonsole

Löschen Sie eine Vorlagensynchronisierungskonfiguration mithilfe der Konsole.

1. Wählen Sie auf der Seite mit den Vorlagendetails die Registerkarte Synchronisieren aus.
2. Wählen Sie im Abschnitt Synchronisierungsdetails die Option Trennen aus.

AWS CLI

Die folgenden Beispielbefehle und Antworten zeigen, wie Sie synchronisierte Vorlagenkonfigurationen mithilfe von löschen können. AWS CLI

Führen Sie den folgenden Befehl aus.

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

Die Antwort lautet wie folgt.

```
{
    "templateSyncConfig": {
        "templateName": "env-template",
        "templateType": "ENVIRONMENT"
    }
}
```

Konfigurationen für die Dienstsynchronisierung

Mit Service Sync können Sie Ihre AWS Proton Dienste mit Git konfigurieren und bereitstellen. Sie können Service Sync verwenden, um erste Bereitstellungen und Updates für Ihren AWS Proton Service mit einer in einem Git-Repository definierten Konfiguration zu verwalten. Über Git kannst du Funktionen wie Versionsverfolgung und Pull-Requests verwenden, um deine Dienste zu konfigurieren, zu verwalten und bereitzustellen. Service Sync kombiniert AWS Proton und Git unterstützt Sie bei der Bereitstellung einer standardisierten Infrastruktur, die über AWS Proton Vorlagen definiert und verwaltet wird. Es verwaltet Dienstdefinitionen in Ihrem Git-Repository und reduziert den Werkzeugwechsel. Im Vergleich zur alleinigen Verwendung von Git können Sie durch die Standardisierung von Vorlagen und Bereitstellung weniger Zeit für die Verwaltung Ihrer Infrastruktur aufwenden. AWS Proton AWS Proton bietet außerdem eine höhere Transparenz und Überprüfbarkeit sowohl für Entwickler als auch für Plattformteams.

AWS Proton OPS-Datei

Die `proton-ops` Datei definiert, wo AWS Proton sich die Spezifikationsdatei befindet, die zum Aktualisieren Ihrer Dienstinstanz verwendet wird. Sie definiert auch, in welcher Reihenfolge

Serviceinstanzen aktualisiert werden sollen und wann Änderungen von einer Instanz auf eine andere übertragen werden sollen.

Die `proton-ops` Datei unterstützt das Synchronisieren einer Dienstinstanz mithilfe der Spezifikationsdatei oder mehrerer Spezifikationsdateien, die sich in Ihrem verknüpften Repository befinden. Sie können dies tun, indem Sie einen Sync-Block in der `proton-ops` Datei definieren, wie im folgenden Beispiel.

Beispiel. /configuration/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
```

Im vorherigen Beispiel sind dies der Dienstname und,`frontend-svc`,,, `alphabeta`, `gamma` und sind die Instanzen. `prod-one` `prod-two` `prod-three`

Bei der `spec` Datei kann es sich um alle Instanzen oder eine Teilmenge der in der `proton-ops` Datei definierten Instanzen handeln. Es muss jedoch mindestens die Instanz innerhalb des Branches und die Spezifikation, mit der sie synchronisiert wird, definiert sein. Wenn in der `proton-ops` Datei keine Instanzen mit dem spezifischen Zweig und dem Speicherort der `spec` Datei definiert sind, erstellt oder aktualisiert Service Sync diese Instanzen nicht.

Die folgenden Beispiele zeigen, wie die `spec` Dateien aussehen. Denken Sie daran, dass die `proton-ops` Datei aus diesen `spec` Dateien synchronisiert wird.

Beispiel: **`./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"
```

Beispiel**`./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"
```

Beispiel**`./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"
```

Wenn eine Instanz nicht synchronisiert wird und beim Versuch, sie zu synchronisieren, weiterhin ein Problem auftritt, kann der Aufruf der `GetServiceInstanceSyncStatus`API zur Lösung des Problems beitragen.

> **Note**
>
> Kunden, die Service Sync verwenden, sind immer noch durch AWS Proton Beschränkungen eingeschränkt.

Blocker

Indem Sie Ihren Service mithilfe AWS Proton von Service Sync synchronisieren, können Sie Ihre Servicespezifikation aktualisieren und Service-Instanzen aus Ihrem Git-Repository erstellen und aktualisieren. Es kann jedoch vorkommen, dass Sie einen Service oder eine Instanz manuell über das AWS-Managementkonsole oder aktualisieren müssen. AWS CLI

AWS Proton verhindert, dass manuelle Änderungen, die Sie über das AWS-Managementkonsole oder vornehmen, überschrieben werden AWS CLI, z. B. das Aktualisieren einer Dienstinstanz oder das Löschen einer Dienstinstanz. Um dies zu erreichen, erstellt es AWS Proton automatisch einen Service Sync-Blocker, indem Service Sync deaktiviert wird, wenn eine manuelle Änderung erkannt wird.

Um alle mit einem Dienst verknüpften Blocker abzurufen, müssen Sie für jeden, der mit dem Dienst `serviceInstance` verknüpft ist, die folgenden Schritte ausführen:

- Rufen Sie die `getServiceSyncBlockerSummary` API nur mit dem auf`serviceName`.
- Rufen Sie die `getServiceSyncBlockerSummary` API mit dem `serviceName` und auf`serviceInstanceName`.

Dadurch wird eine Liste der neuesten Blocker und der ihnen zugeordnete Status zurückgegeben. Wenn irgendwelche Blocker als AKTIV markiert sind, müssen Sie sie lösen, indem Sie die `UpdateServiceSyncBlocker` API mit dem `blockerId` und `resolvedReason` für jeden Block aufrufen.

Wenn Sie eine Dienstinstanz manuell aktualisieren oder erstellen, AWS Proton wird auf der Dienstinstanz ein Dienstsynchronisierungsblocker erstellt. AWS Proton synchronisiert weiterhin alle anderen Dienstinstanzen, deaktiviert jedoch die Synchronisierung dieser Dienstinstanz, bis der Blocker behoben ist. Wenn Sie eine Dienstinstanz aus einem Dienst löschen, AWS Proton wird für den Dienst ein Dienstsynchronisierungsblocker erstellt. Dadurch wird AWS Proton verhindert, dass eine der Dienstinstanzen synchronisiert wird, bis der Blocker behoben wurde.

Wenn Sie alle aktiven Blocker haben, müssen Sie sie lösen, indem Sie die `UpdateServiceSyncBlocker` API mit `blockerId` und `resolvedReason` für jeden der aktiven Blocker aufrufen.

Mithilfe von können Sie feststellen AWS-Managementkonsole, ob eine Dienstsynchronisierung deaktiviert ist, indem Sie zur Registerkarte Service Sync navigieren AWS Proton und diese auswählen. Wenn der Dienst oder die Dienstinstanzen blockiert sind, wird die Schaltfläche Aktivieren angezeigt. Um die Dienstsynchronisierung zu aktivieren, wählen Sie Aktivieren.

Topics

Erstellen Sie eine Service Sync-Konfiguration

Sie können eine Service Sync-Konfiguration mithilfe der Konsole oder erstellen AWS CLI.

AWS-Managementkonsole

1. Wählen Sie auf der Seite Servicevorlage auswählen eine Vorlage aus und klicken Sie auf Konfigurieren.
2. Geben Sie auf der Seite Dienst konfigurieren im Abschnitt Dienstdetails einen neuen Dienstnamen ein.
3. (Optional) Geben Sie eine Beschreibung für den Dienst ein.
4. Wählen Sie im Abschnitt Quellcode-Repository der Anwendung die Option Verlinktes Git-Repository auswählen aus, um ein Repository auszuwählen, mit dem Sie bereits verknüpft sind AWS Proton. Wenn du noch kein verlinktes Repository hast, wähle Ein anderes Git-Repository verknüpfen und befolge die Anweisungen unter Link zu deinem Repository erstellen.
5. Wählen Sie für Repository den Namen Ihres Quellcode-Repositorys aus der Liste aus.
6. Wählen Sie für Branch den Namen des Repository-Branches für Ihren Quellcode aus der Liste aus.
7. (Optional) Wählen Sie im Abschnitt „Tags“ die Option Neues Tag hinzufügen aus und geben Sie einen Schlüssel und einen Wert ein, um ein vom Kunden verwaltetes Tag zu erstellen.
8. Wählen Sie Weiter aus.
9. Wählen Sie auf der Seite Dienstinstanzen konfigurieren im Abschnitt Service-Definitionsquelle die Option Service mit Git synchronisieren aus.
10. Wenn Sie Ihre `proton-ops` Datei erstellen möchten AWS Proton , wählen Sie im Abschnitt Servicedefinitionsdateien die Option Ich möchte, dass AWS Proton die Dateien erstellt. Mit dieser Option wird die `proton-ops` AND-Datei an `spec` den von Ihnen angegebenen Speicherorten AWS Proton erstellt. Wählen Sie Ich stelle meine eigenen Dateien bereit, um Ihre eigene OPS-Datei zu erstellen.
11. Wählen Sie im Abschnitt Service-Definition-Repository die Option Ein verknüpftes Git-Repository auswählen aus, um ein Repository auszuwählen, mit dem Sie bereits verknüpft sind AWS Proton.
12. Wählen Sie unter Repository-Name den Namen Ihres Quellcode-Repositorys aus der Liste aus.
13. Wählen Sie für den **`proton-ops`**Dateizweig den Namen Ihres Zweigs aus der Liste aus, in der Ihre OPS- und Spezifikationsdatei gespeichert AWS Proton werden soll.

14. Im Abschnitt Serviceinstanzen wird jedes Feld automatisch auf der Grundlage der Werte in der `proton-ops` Datei gefüllt.
15. Wählen Sie Weiter und überprüfen Sie Ihre Eingaben.
16. Wählen Sie Erstellen aus.

AWS CLI

Erstellen Sie eine Service Sync-Konfiguration mit dem AWS CLI

- Führen Sie den folgenden Befehl aus.

```
$ 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)
```

Die Antwort lautet wie folgt.

```
{
    "serviceSyncConfig": {
        "branch": "main",
        "filePath": "./configuration/custom-proton-ops.yaml",
        "repositoryName": "example/proton-sync-service",
        "repositoryProvider": "GITHUB",
        "serviceName": "service name"
    }
}
```

Konfigurationsdetails für eine Dienstsynchronisierung anzeigen

Sie können die Konfigurationsdetails für eine Servicesynchronisierung in der Konsole oder anzeigen AWS CLI.

AWS-Managementkonsole

Verwenden Sie die Konsole, um die Konfigurationsdetails für eine Servicesynchronisierung anzuzeigen

1. Wählen Sie im Navigationsbereich Services.
2. Um Detaildaten anzuzeigen, wählen Sie den Namen eines Dienstes aus, für den Sie eine Service Sync-Konfiguration erstellt haben.
3. Wählen Sie auf der Detailseite für den Service die Registerkarte Service Sync aus, um die Konfigurationsdetaildaten für die Service Sync anzuzeigen.

AWS CLI

Verwenden Sie den AWS CLI , um einen synchronisierten Dienst abzurufen.

Führen Sie den folgenden Befehl aus.

```
$ aws proton get-service-sync-config \
    --service-name "service name"
```

Die Antwort lautet wie folgt.

```
{
    "serviceSyncConfig": {
        "branch": "main",
        "filePath": "./configuration/custom-proton-ops.yaml",
        "repositoryName": "example/proton-sync-service",
        "repositoryProvider": "GITHUB",
        "serviceName": "service name"
    }
}
```

Verwenden Sie den AWS CLI , um den Status der Dienstsynchronisierung abzurufen.

Führen Sie den folgenden Befehl aus.

```
$ aws proton get-service-sync-status \
    --service-name "service name"
```

Bearbeiten Sie eine Dienstsynchronisierungskonfiguration

Sie können eine Service Sync-Konfiguration mit der Konsole oder bearbeiten AWS CLI.

AWS-Managementkonsole

Bearbeiten Sie eine Service Sync-Konfiguration mithilfe der Konsole.

1. Wählen Sie im Navigationsbereich Services.
2. Um Detaildaten anzuzeigen, wählen Sie den Namen eines Dienstes aus, für den Sie eine Service Sync-Konfiguration erstellt haben.
3. Wählen Sie auf der Service-Detailseite die Registerkarte Service Sync aus.
4. Wählen Sie im Abschnitt Service Sync die Option Bearbeiten aus.
5. Aktualisieren Sie auf der Seite Bearbeiten die Informationen, die Sie bearbeiten möchten, und wählen Sie dann Speichern.

AWS CLI

Der folgende Beispielbefehl und die folgende Antwort zeigen, wie Sie eine Service Sync-Konfiguration mit dem bearbeiten können AWS CLI.

Führen Sie den folgenden Befehl aus.

```
$ 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"
```

Die Antwort lautet wie folgt.

```
{
    "serviceSyncConfig": {
        "branch": "main",
        "filePath": "./configuration/custom-proton-ops.yaml",
        "repositoryName": "example/proton-sync-service",
```

```
        "repositoryProvider": "GITHUB",
        "serviceName": "service name"
    }
}
```

Löschen Sie eine Service Sync-Konfiguration

Sie können eine Service Sync-Konfiguration über die Konsole oder löschen AWS CLI.

AWS-Managementkonsole

Löschen Sie eine Service Sync-Konfiguration mithilfe der Konsole

1. Wählen Sie auf der Seite mit den Dienstdetails die Registerkarte Service Sync aus.
2. Wählen Sie im Abschnitt Details zur Servicesynchronisierung die Option Verbindung trennen, um die Verbindung zu Ihrem Repository zu trennen. Nachdem dein Repository getrennt wurde, synchronisieren wir den Service nicht mehr mit diesem Repository.

AWS CLI

Die folgenden Beispielbefehle und Antworten zeigen, wie Sie mit dem Dienst synchronisierte Konfigurationen löschen AWS CLI können.

Führen Sie den folgenden Befehl aus.

```
$ aws proton delete-service-sync-config \
    --service-name "service name"
```

Die Antwort lautet wie folgt.

```
{
    "serviceSyncConfig": {
        "branch": "main",
        "filePath": "./configuration/custom-proton-ops.yaml",
        "repositoryName": "example/proton-sync-service",
        "repositoryProvider": "GITHUB",
        "serviceName": "service name"
    }
}
```

> **Note**
>
> Service Sync löscht keine Dienstinstanzen. Es löscht nur die Konfiguration.

AWS Proton Umgebungen

Denn AWS Proton eine Umgebung stellt den Satz gemeinsam genutzter Ressourcen und Richtlinien dar, in denen AWS Proton Dienste bereitgestellt werden. Sie können alle Ressourcen enthalten, von denen erwartet wird, dass sie von allen AWS Proton Serviceinstanzen gemeinsam genutzt werden. Zu VPCs diesen Ressourcen können Cluster und gemeinsam genutzte Load Balancer oder API-Gateways gehören. Eine AWS Proton Umgebung muss erstellt werden, bevor ein Dienst in ihr bereitgestellt werden kann.

In diesem Abschnitt wird beschrieben, wie Umgebungen mithilfe von Erstellungs-, Anzeige-, Aktualisierungs- und Löschvorgängen verwaltet werden. Weitere Informationen finden Sie in der Referenz zur AWS Proton Service-API.

Themen

IAM-Rollen

Mit AWS Proton stellen Sie die IAM-Rollen und AWS KMS -Schlüssel für die AWS Ressourcen bereit, die Sie besitzen und verwalten. Diese werden später auf Ressourcen angewendet und von diesen genutzt, die Entwicklern gehören und von diesen verwaltet werden. Sie erstellen eine IAM-Rolle, um den Zugriff Ihres Entwicklerteams auf die AWS Proton API zu kontrollieren.

AWS Proton Servicerolle

Wenn Sie eine neue Umgebung erstellen, geben Sie eine zugehörige IAM-Servicerolle an. Die Rolle enthält alle Berechtigungen, die erforderlich sind, um die gesamte bereitgestellte Infrastruktur zu

aktualisieren, die sowohl in den Umgebungsvorlagen als auch in den Dienstvorlagen definiert ist. Rollenbeispiele finden Sie unterAWS Proton Servicerolle für die Bereitstellung mit CloudFormation. Wenn Sie Verbindungen mit Umgebungskonten und Umgebungskonten verwenden, erstellen Sie die Rolle in einem ausgewählten Umgebungskonto. Weitere Informationen erhalten Sie unter Erstellen Sie eine Umgebung in einem Konto und stellen Sie sie in einem anderen Konto bereit und Verbindungen zu Umgebungskonten.

Wie Sie diese Servicerolle bereitstellen und wer die Rolle übernimmt, hängt von der Bereitstellungsmethode Ihrer Umgebung ab.

- AWS-verwaltete Bereitstellung — Sie stellen die Rolle entweder direkt beim Erstellen einer Umgebung oder indirekt über Kontoverbindungen bereit. AWS Proton AWS Proton übernimmt die Rolle im entsprechenden Konto für die Bereitstellung der Umgebung und der Serviceinfrastruktur.
- Selbstverwaltete Bereitstellung — Es liegt in Ihrer Verantwortung, Ihre Provisioning-Automatisierung so zu konfigurieren, dass sie mithilfe der entsprechenden Anmeldeinformationen eine entsprechende Rolle einnimmt, wenn eine Pull-Anfrage (PR) eine Bereitstellungsaktion auslöst. Ein Beispiel für eine GitHub Aktion, die eine Rolle annimmt, finden Sie unter Eine Rolle annehmen in der Dokumentation Aktion für Aktionen „ AWS Anmeldeinformationen konfigurieren“. GitHub

Weitere Informationen zu Bereitstellungsmethoden finden Sie unterthe section called “Bereitstellungsmethoden”.

Erstellen einer Umgebung

Lernen Sie, AWS Proton Umgebungen zu erstellen.

Sie können eine AWS Proton Umgebung auf zwei Arten erstellen:

- Mithilfe einer Standardumgebungsvorlage können Sie eine Standardumgebung erstellen, verwalten und bereitstellen. AWS Proton stellt die Infrastruktur für Ihre Umgebung bereit.
- Stellen Sie mithilfe einer AWS Proton Vorlage für eine vom Kunden verwaltete Umgebung eine Connect zur vom Kunden verwalteten Infrastruktur her. Sie stellen Ihre eigenen gemeinsam genutzten Ressourcen außerhalb von bereit und stellen dann Bereitstellungsausgaben bereit AWS Proton, die AWS Proton verwendet werden können.

Beim Erstellen einer Umgebung können Sie einen von mehreren Bereitstellungsansätzen wählen.

- AWS verwaltete Bereitstellung — Erstellen, verwalten und bereitstellen Sie eine Umgebung in einem einzigen Konto. AWS Proton stellt Ihre Umgebung bereit.

  Diese Methode unterstützt nur Vorlagen CloudFormation für Infrastrukturcode (IaC).

- AWS verwaltete Bereitstellung für ein anderes Konto — Erstellen und verwalten Sie in einem einzigen Verwaltungskonto eine Umgebung, die in einem anderen Konto mit Verbindungen zu Umgebungskonten bereitgestellt wird. AWS Proton stellt Ihre Umgebung in dem anderen Konto bereit. Weitere Informationen erhalten Sie unter Erstellen Sie eine Umgebung in einem Konto und stellen Sie sie in einem anderen Konto bereit und Verbindungen zu Umgebungskonten.

  Diese Methode unterstützt nur CloudFormation IaC-Vorlagen.

- Selbstverwaltete Bereitstellung — AWS Proton sendet Bereitstellungs-Pull-Anfragen an ein verknüpftes Repository mit Ihrer eigenen Bereitstellungsinfrastruktur.

  Diese Methode unterstützt nur Terraform IaC-Vorlagen.

- CodeBuild Bereitstellung — AWS Proton verwendet, AWS CodeBuild um von Ihnen bereitgestellte Shell-Befehle auszuführen. Ihre Befehle können Eingaben lesen, die die Infrastruktur bereitstellen, und sind für die AWS Proton Bereitstellung oder Deprovisionierung der Infrastruktur und die Generierung von Ausgabewerten verantwortlich. Ein Vorlagenpaket für diese Methode enthält Ihre Befehle in einer Manifestdatei sowie alle Programme, Skripts oder anderen Dateien, die diese Befehle möglicherweise benötigen.

  Als Beispiel für die Verwendung von CodeBuild Provisioning können Sie Code hinzufügen, der die AWS Cloud Development Kit (AWS CDK) zur Bereitstellung von AWS Ressourcen verwendet, und ein Manifest, das das CDK installiert und Ihren CDK-Code ausführt.

  Weitere Informationen finden Sie unter the section called “CodeBuild bündeln”.

  > **Note**
  >
  > Sie können die CodeBuild Bereitstellung mit Umgebungen und Diensten verwenden. Derzeit können Sie Komponenten auf diese Weise nicht bereitstellen.

Bei der AWS verwalteten Bereitstellung (sowohl für dasselbe Konto als auch für ein anderes Konto) werden AWS Proton direkte Aufrufe zur Bereitstellung Ihrer Ressourcen getätigt.

Bei der selbstverwalteten Bereitstellung werden Pull-Anfragen gestellt AWS Proton , um kompilierte IaC-Dateien bereitzustellen, die Ihre IaC-Engine zur Bereitstellung von Ressourcen verwendet.

Weitere Informationen finden Sie unter the section called “Bereitstellungsmethoden”, the section called “Vorlagenpakete” und the section called “Anforderungen an das Umgebungsschema”.

Topics

Erstellen Sie eine Standardumgebung und stellen Sie sie in demselben Konto bereit

Verwenden Sie die Konsole oder AWS CLI um eine Umgebung in einem einzigen Konto zu erstellen und bereitzustellen. Die Bereitstellung wird verwaltet von AWS.

AWS-Managementkonsole

Verwenden Sie die Konsole, um eine Umgebung in einem einzigen Konto zu erstellen und bereitzustellen

1. Wählen Sie in der AWS Proton Konsole Umgebungen aus.
2. Wählen Sie Create environment (Umgebung erstellen) aus.
3. Wählen Sie auf der Seite Umgebungsvorlage auswählen eine Vorlage aus und klicken Sie auf Konfigurieren.
4. Wählen Sie auf der Seite Umgebung konfigurieren im Abschnitt Provisioning die Option AWS Managed Provisioning aus.
5. Wählen Sie im Abschnitt Bereitstellungskonto die Option Dies aus. AWS-Konto
6. Geben Sie auf der Seite Umgebung konfigurieren im Abschnitt Umgebungseinstellungen einen Umgebungsnamen ein.
7. (Optional) Geben Sie eine Beschreibung für die Umgebung ein.
8. Wählen Sie im Abschnitt Umgebungsrollen die AWS Proton Servicerolle aus, die Sie als Teil von erstellt habenAWS Proton Servicerollen einrichten.
9. (Optional) Wählen Sie im Abschnitt Komponentenrolle eine Servicerolle aus, die die Ausführung direkt definierter Komponenten in der Umgebung ermöglicht und den Umfang der

Ressourcen, die sie bereitstellen können, begrenzt. Weitere Informationen finden Sie unter Komponenten.

10. (Optional) Wählen Sie im Abschnitt „Tags“ die Option Neues Tag hinzufügen aus und geben Sie einen Schlüssel und einen Wert ein, um ein vom Kunden verwaltetes Tag zu erstellen.
11. Wählen Sie Weiter aus.
12. Auf der Seite Benutzerdefinierte Umgebungseinstellungen konfigurieren müssen Sie Werte für die `required` Parameter eingeben. Sie können Werte für die `optional` Parameter eingeben oder die Standardwerte verwenden, sofern diese angegeben sind.
13. Wählen Sie Weiter und überprüfen Sie Ihre Eingaben.
14. Wählen Sie Erstellen aus.

    Sehen Sie sich die Umgebungsdetails und den Status sowie die AWS verwalteten und vom Kunden verwalteten Tags für Ihre Umgebung an.
15. Wählen Sie im Navigationsbereich Environments (Umgebungen) aus.

    Auf einer neuen Seite wird eine Liste Ihrer Umgebungen zusammen mit dem Status und anderen Umgebungsdetails angezeigt.

AWS CLI

Verwenden Sie die AWS CLI , um eine Umgebung in einem einzigen Konto zu erstellen und bereitzustellen.

Um eine Umgebung zu erstellen, geben Sie den ARN der AWS Proton Dienstrolle, den Pfad zu Ihrer Spezifikationsdatei, den Umgebungsnamen, den ARN der Umgebungsvorlage, die Haupt- und Nebenversionen sowie eine Beschreibung (optional) an.

Die nächsten Beispiele zeigen eine YAML formatierte Spezifikationsdatei, die Werte für zwei Eingaben angibt, die in der Schemadatei der Umgebungsvorlage definiert sind. Sie können den `get-environment-template-minor-version` Befehl verwenden, um das Umgebungsvorlagenschema anzuzeigen.

```
proton: EnvironmentSpec
spec:
  my_sample_input: "the first"
  my_other_sample_input: "the second"
```

Erstellen Sie eine Umgebung, indem Sie den folgenden Befehl ausführen.

```
$ 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"
```

Antwort:

```
{
    "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"
    }
}
```

Nachdem Sie eine neue Umgebung erstellt haben, können Sie eine Liste mit AWS und vom Kunden verwalteten Tags anzeigen, wie im folgenden Beispielbefehl gezeigt. AWS Proton generiert automatisch AWS verwaltete Tags für Sie. Mit dem können Sie auch vom Kunden verwaltete Tags ändern und erstellen AWS CLI. Weitere Informationen finden Sie unter AWS Proton Ressourcen und Tagging.

Befehl:

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

Erstellen Sie eine Umgebung in einem Konto und stellen Sie sie in einem anderen Konto bereit

Verwenden Sie die Konsole oder AWS CLI um eine Standardumgebung in einem Verwaltungskonto zu erstellen, das die Umgebungsinfrastruktur in einem anderen Konto bereitstellt. Die Bereitstellung wird verwaltet von AWS.

Führen Sie die folgenden Schritte aus, bevor Sie die Konsole oder CLI verwenden.

1. Identifizieren Sie das AWS-Konto IDs Verwaltungs- und das Umgebungskonto und kopieren Sie sie für die spätere Verwendung.
2. Erstellen Sie im Umgebungskonto eine AWS Proton Servicerolle mit Mindestberechtigungen für die zu erstellende Umgebung. Weitere Informationen finden Sie unter AWS Proton Servicerolle für die Bereitstellung mit CloudFormation.

AWS-Managementkonsole

Verwenden Sie die Konsole, um eine Umgebung in einem Konto zu erstellen und in einem anderen bereitzustellen.

1. Erstellen Sie im Umgebungskonto eine Verbindung mit dem Umgebungskonto und senden Sie damit eine Anfrage zur Verbindung mit dem Verwaltungskonto.

   a. Wählen Sie in der AWS Proton Konsole im Navigationsbereich die Option Environment account connections aus.

   b. Wählen Sie auf der Seite Verbindungen zu Umgebungskonten die Option Verbindungsanfrage aus.

   > **Note**
   >
   > Vergewissern Sie sich, dass die Konto-ID, die in der Überschrift der Seite „Verbindung mit dem Umgebungskonto“ aufgeführt ist, mit Ihrer vorab identifizierten Umgebungskonto-ID übereinstimmt.

   c. Wählen Sie auf der Seite Verbindungsanfrage im Abschnitt Umgebungsrolle die Option Bestehende Servicerolle und den Namen der Servicerolle aus, die Sie für die Umgebung erstellt haben.

   d. Geben Sie im Abschnitt Mit Verwaltungskonto verbinden die Verwaltungskonto-ID und einen Umgebungsnamen für Ihre AWS Proton Umgebung ein. Kopieren Sie den Namen für die spätere Verwendung.

   e. Wählen Sie in der unteren rechten Ecke der Seite die Option Verbindungsanfrage aus.

   f. Ihre Anfrage wird in der Tabelle Environment-Verbindungen, die an ein Verwaltungskonto gesendet wurden, als ausstehend angezeigt. In einem Modalfenster wird gezeigt, wie Sie die Anfrage vom Verwaltungskonto annehmen können.

2. Akzeptieren Sie im Verwaltungskonto eine Verbindungsanfrage vom Umgebungskonto aus.

   a. Melden Sie sich bei Ihrem Verwaltungskonto an und wählen Sie in der AWS Proton Konsole Environment Account Connections aus.

   b. Wählen Sie auf der Seite Verbindungen zu Umgebungskonten in der Tabelle Verbindungsanfragen für Umgebungskonten die Umgebungskontoverbindung mit der Umgebungskonto-ID aus, die Ihrer vorab identifizierten Umgebungskonto-ID entspricht.

   > **Note**
   >
   > Vergewissern Sie sich, dass die Konto-ID, die in der Überschrift Verbindungsseite für das Umweltkonto aufgeführt ist, mit Ihrer vorab identifizierten Verwaltungskonto-ID übereinstimmt.

   c. Wählen Sie Accept (Akzeptieren) aus. Der Status ändert sich von PENDING zu CONNECTED.

3. Erstellen Sie im Verwaltungskonto eine Umgebung.

   a. Wählen Sie im Navigationsbereich die Option Umgebungsvorlagen aus.

   b. Wählen Sie auf der Seite Umgebungsvorlagen die Option Umgebungsvorlage erstellen aus.

   c. Wählen Sie auf der Seite Umgebungsvorlage auswählen eine Umgebungsvorlage aus.

   d. Wählen Sie auf der Seite Umgebung konfigurieren im Abschnitt Provisioning die Option AWS Managed Provisioning aus.

   e. Wählen Sie im Abschnitt Bereitstellungskonto die Option Anderes AWS Konto; aus.

   f. Wählen Sie im Abschnitt Umgebungsdetails die Verbindung Ihres Umgebungskontos und den Umgebungsnamen aus.

   g. Wählen Sie Weiter aus.

   h. Füllen Sie die Formulare aus und klicken Sie auf Weiter, bis Sie zur Seite Überprüfen und Erstellen gelangen.

   i. Überprüfen Sie und wählen Sie Umgebung erstellen.

AWS CLI

Verwenden Sie die AWS CLI , um eine Umgebung in einem Konto zu erstellen und in einem anderen bereitzustellen.

Erstellen Sie im Umgebungskonto eine Verbindung mit dem Umgebungskonto und fordern Sie die Verbindung an, indem Sie den folgenden Befehl ausführen.

```
$ 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"
```

Antwort:

```
{
    "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"
    }
}
```

Akzeptieren Sie im Verwaltungskonto die Verbindungsanfrage für das Umgebungskonto, indem Sie den folgenden Befehl ausführen.

```
$ aws proton accept-environment-account-connection \
    --id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

Antwort:

```
{
    "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"
    }
}
```

Zeigen Sie die Verbindung Ihres Umgebungskontos an, indem Sie den folgenden Befehl ausführen.

```
$ aws proton get-environment-account-connection \
    --id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

Antwort:

```
{
    "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"
    }
}
```

Erstellen Sie im Verwaltungskonto eine Umgebung, indem Sie den folgenden Befehl ausführen.

```
$ 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"
```

Antwort:

```
{
    "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"
    }
}
```

Erstellen Sie eine Umgebung mithilfe von selbstverwalteter Bereitstellung und stellen Sie sie bereit

Wenn Sie die selbstverwaltete Bereitstellung verwenden, werden AWS Proton Bereitstellungs-Pull-Requests an ein verknüpftes Repository mit Ihrer eigenen Bereitstellungsinfrastruktur gesendet. Die Pull Requests starten Ihren eigenen Workflow, der AWS Dienste aufruft, um die Infrastruktur bereitzustellen.

Überlegungen zur selbstverwalteten Bereitstellung:

- Bevor Sie eine Umgebung erstellen, richten Sie ein Repository-Ressourcenverzeichnis für die selbstverwaltete Bereitstellung ein. Weitere Informationen finden Sie unter AWS Proton Infrastruktur als Codedateien.
- AWS Proton Wartet nach dem Erstellen der Umgebung auf asynchrone Benachrichtigungen zum Status Ihrer Infrastrukturbereitstellung. Ihr Bereitstellungscode muss die AWS Proton `NotifyResourceStateChange` API verwenden, an die diese asynchronen Benachrichtigungen gesendet werden sollen. AWS Proton

Sie können die selbstverwaltete Bereitstellung in der Konsole oder mit dem verwenden. AWS CLI Die folgenden Beispiele zeigen, wie Sie die selbstverwaltete Bereitstellung mit Terraform verwenden können.

AWS-Managementkonsole

Verwenden Sie die Konsole, um mithilfe von selbstverwalteter Bereitstellung eine Terraform-Umgebung zu erstellen.

1. Wählen Sie in der AWS Proton Konsole Umgebungen aus.
2. Wählen Sie Create environment (Umgebung erstellen) aus.
3. Wählen Sie auf der Seite Umgebungsvorlage auswählen eine Terraform-Vorlage aus und klicken Sie auf Konfigurieren.
4. Wählen Sie auf der Seite Umgebung konfigurieren im Abschnitt Provisioning die Option Selbstverwaltete Bereitstellung aus.
5. Gehen Sie im Abschnitt Details zum Bereitstellungs-Repository wie folgt vor:

   a. Wenn Sie Ihr Provisioning-Repository noch nicht mit verknüpft haben AWS Proton, wählen Sie Neues Repository, wählen Sie einen der Repository-Anbieter und dann als CodeStarVerbindung eine Ihrer Verbindungen aus.

   > **Note**
   >
   > Wenn Sie noch keine Verbindung zum entsprechenden Repository-Anbieter-Konto haben, wählen Sie Neue CodeStar Verbindung hinzufügen aus. Stellen Sie dann eine Verbindung her und klicken Sie dann auf die Schaltfläche „Aktualisieren“ neben dem CodeStar Verbindungsmenü. Sie sollten jetzt in der Lage sein, Ihre neue Verbindung im Menü auszuwählen.

   Wenn du dein Repository bereits mit verknüpft hast AWS Proton, wähle Bestehendes Repository aus.

   b. Wählen Sie als Repository-Name ein Repository aus. Im Dropdownmenü werden verknüpfte Repositorys für Bestehendes Repository oder die Liste der Repositorys im Anbieterkonto für Neues Repository angezeigt.

   c. Wählen Sie als Branch-Name einen der Repository-Branches aus.
6. Geben Sie im Abschnitt Umgebungseinstellungen einen Umgebungsnamen ein.
7. (Optional) Geben Sie eine Beschreibung für die Umgebung ein.
8. (Optional) Wählen Sie im Abschnitt „Tags“ die Option Neues Tag hinzufügen aus und geben Sie einen Schlüssel und einen Wert ein, um ein vom Kunden verwaltetes Tag zu erstellen.

9. Wählen Sie Weiter aus.
10. Auf der Seite Benutzerdefinierte Umgebungseinstellungen konfigurieren müssen Sie Werte für die `required` Parameter eingeben. Sie können Werte für die `optional` Parameter eingeben oder die Standardwerte verwenden, sofern diese angegeben sind.
11. Wählen Sie Weiter und überprüfen Sie Ihre Eingaben.
12. Wähle Erstellen, um eine Pull-Anfrage zu senden.

    - Wenn Sie die Pull-Anfrage genehmigen, ist die Bereitstellung im Gange.
    - Wenn Sie die Pull-Anfrage ablehnen, wird die Erstellung der Umgebung abgebrochen.
    - Wenn bei der Pull-Anfrage ein Timeout auftritt, ist die Erstellung der Umgebung nicht abgeschlossen.
13. Sehen Sie sich die Umgebungsdetails und den Status sowie die AWS verwalteten und vom Kunden verwalteten Tags für Ihre Umgebung an.
14. Wählen Sie im Navigationsbereich Environments (Umgebungen) aus.

    Auf einer neuen Seite wird eine Liste Ihrer Umgebungen zusammen mit dem Status und anderen Umgebungsdetails angezeigt.

AWS CLI

Wenn Sie eine Umgebung mithilfe von Self-Managed Provisioning erstellen, fügen Sie den `provisioningRepository` Parameter hinzu und lassen die Parameter und weg. `ProtonServiceRoleArn` `environmentAccountConnectionId`

Verwenden Sie den AWS CLI , um eine Terraform-Umgebung mit selbstverwalteter Bereitstellung zu erstellen.

1. Erstellen Sie eine Umgebung und senden Sie eine Pull-Anfrage zur Überprüfung und Genehmigung an das Repository.

   Das nächste Beispiel zeigt eine YAML formatierte Spezifikationsdatei, die die Werte für zwei Eingaben auf der Grundlage der Schemadatei der Umgebungsvorlage definiert. Sie können den `get-environment-template-minor-version` Befehl verwenden, um das Umgebungsvorlagenschema anzuzeigen.

   Spezifikation:

```
proton: EnvironmentSpec
spec:
  ssm_parameter_value: "test"
```

Erstellen Sie eine Umgebung, indem Sie den folgenden Befehl ausführen.

```
$ 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"
```

Antwort: >

```
{
    "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. Überprüfen Sie die Anfrage.

   - Wenn Sie die Anfrage genehmigen, ist die Bereitstellung im Gange.
   - Wenn Sie die Anfrage ablehnen, wird die Erstellung der Umgebung abgebrochen.
   - Wenn bei der Pull-Anfrage ein Timeout auftritt, ist die Erstellung der Umgebung nicht abgeschlossen.

3. Geben Sie asynchron den Bereitstellungsstatus an. AWS Proton Das folgende Beispiel informiert über eine erfolgreiche AWS Proton Bereitstellung.

```
$ aws proton notify-resource-deployment-status-change \
    --resource-arn "arn:aws:proton:region-id:123456789012:environment/pr-environment" \
    --status "SUCCEEDED"
```

Umgebungsdaten anzeigen

Sie können Umgebungsdetaildaten entweder mit der AWS Proton Konsole oder mit dem anzeigen AWS CLI.

AWS-Managementkonsole

Mithilfe der AWS Proton Konsole können Sie Listen von Umgebungen mit Details und einzelne Umgebungen mit Detaildaten anzeigen.

1. Um eine Liste Ihrer Umgebungen anzuzeigen, wählen Sie im Navigationsbereich Umgebungen aus.
2. Um Detaildaten anzuzeigen, wählen Sie den Namen einer Umgebung aus.

   Zeigen Sie die Detaildaten Ihrer Umgebung an.

AWS CLI

Verwenden Sie die AWS CLI Option Umgebungsdetails abrufen oder auflisten.

Führen Sie den folgenden Befehl aus:

```
$ aws proton get-environment \
    --name "MySimpleEnv"
```

Antwort:

```
{
    "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"
    }
}
```

Aktualisieren einer Umgebung

Wenn die AWS Proton Umgebung mit einer Umgebungskontoverbindung verknüpft ist, aktualisieren Sie den `protonServiceRoleArn` Parameter nicht und fügen Sie ihn nicht hinzu, um eine Verbindung mit einem Umgebungskonto zu aktualisieren oder eine Verbindung herzustellen.

Sie können nur auf eine neue Umgebungskontoverbindung aktualisieren, wenn beide der folgenden Bedingungen zutreffen:

- Die Umgebungskontoverbindung wurde in demselben Umgebungskonto erstellt, in dem die aktuelle Umgebungskontoverbindung erstellt wurde.
- >Die Verbindung mit dem Umgebungskonto ist mit der aktuellen Umgebung verknüpft.

Wenn die Umgebung keiner Verbindung mit einem Umgebungskonto zugeordnet ist, aktualisieren Sie den `environmentAccountConnectionId` Parameter nicht und fügen Sie ihn nicht hinzu.

Sie können entweder den `protonServiceRoleArn` Parameter `environmentAccountConnectionId` oder und den Wert aktualisieren. Sie können nicht beide aktualisieren.

Wenn Ihre Umgebung selbstverwaltete Bereitstellung verwendet, aktualisieren Sie den Parameter nicht und lassen Sie die `provisioning-repository` Parameter und weg. `environmentAccountConnectionId` `protonServiceRoleArn`

Es gibt vier Modi für die Aktualisierung einer Umgebung, wie in der folgenden Liste beschrieben. Bei Verwendung von definiert das `deployment-type` Feld den Modus. AWS CLI Bei Verwendung

der Konsole werden diese Modi den Aktionen Bearbeiten, Aktualisieren, Nebenaktualisierung und Hauptaktualisierung zugeordnet, die im Dropdownmenü Aktionen angezeigt werden.

`NONE`

In diesem Modus findet keine Bereitstellung statt. Nur die angeforderten Metadatenparameter werden aktualisiert.

`CURRENT_VERSION`

In diesem Modus wird die Umgebung bereitgestellt und mit der neuen Spezifikation, die Sie angeben, aktualisiert. Nur die angeforderten Parameter werden aktualisiert. Geben Sie keine Parameter für Neben- oder Hauptversionen an, wenn Sie dies verwenden`deployment-type`.

`MINOR_VERSION`

In diesem Modus wird die Umgebung standardmäßig mit der veröffentlichten, empfohlenen (neuesten) Nebenversion der aktuell verwendeten Hauptversion bereitgestellt und aktualisiert. Sie können auch eine andere Nebenversion der aktuell verwendeten Hauptversion angeben.

`MAJOR_VERSION`

In diesem Modus wird die Umgebung standardmäßig mit der veröffentlichten, empfohlenen (neuesten) Haupt- und Nebenversion der aktuellen Vorlage bereitgestellt und aktualisiert. Sie können auch eine andere Hauptversion angeben, die höher ist als die verwendete Hauptversion und eine Nebenversion (optional).

Topics

Aktualisieren Sie eine AWS verwaltete Bereitstellungsumgebung

Die Standardbereitstellung wird nur von Umgebungen unterstützt, in denen die Bereitstellung mit erfolgt. CloudFormation

Verwenden Sie die Konsole oder aktualisieren AWS CLI Sie Ihre Umgebung.

AWS-Managementkonsole

Aktualisieren Sie eine Umgebung mithilfe der Konsole, wie in den folgenden Schritten gezeigt.

1. Wählen Sie einen der folgenden 2 Schritte aus.
   a. In der Liste der Umgebungen.
      i. Wählen Sie in der AWS Proton Konsole Umgebungen aus.
      ii. Wählen Sie in der Liste der Umgebungen das Optionsfeld links neben der Umgebung aus, die Sie aktualisieren möchten.
   b. Auf der Detailseite der Konsolenumgebung.
      i. Wählen Sie in der AWS Proton Konsole Umgebungen aus.
      ii. Wählen Sie in der Liste der Umgebungen den Namen der Umgebung aus, die Sie aktualisieren möchten.
2. Wählen Sie einen der nächsten 4 Schritte aus, um Ihre Umgebung zu aktualisieren.
   a. Um eine Änderung vorzunehmen, für die keine Bereitstellung der Umgebung erforderlich ist.
      i. Zum Beispiel, um eine Beschreibung zu ändern.

         Wählen Sie Bearbeiten aus.
      ii. Füllen Sie das Formular aus und wählen Sie Weiter.
      iii. Überprüfe deine Bearbeitung und wähle „Aktualisieren“.
   b. Um nur die Metadateneingaben zu aktualisieren.
      i. Wählen Sie Aktionen und dann Aktualisieren aus.
      ii. Füllen Sie das Formular aus und wählen Sie Bearbeiten.
      iii. Füllen Sie die Formulare aus und wählen Sie Weiter, bis Sie zur Seite „Überprüfen“ gelangen.
      iv. Überprüfen Sie Ihre Aktualisierungen und wählen Sie Aktualisieren.
   c. Um eine neue Nebenversion der zugehörigen Umgebungsvorlage zu aktualisieren.
      i. Wählen Sie „Aktionen“ und dann „Nebenprogramm aktualisieren“.

ii. Füllen Sie das Formular aus und wählen Sie Weiter.

iii. Füllen Sie die Formulare aus und wählen Sie Weiter, bis Sie zur Seite „Überprüfen“ gelangen.

iv. Überprüfen Sie Ihre Aktualisierungen und wählen Sie Aktualisieren.

d. Um eine neue Hauptversion der zugehörigen Umgebungsvorlage zu aktualisieren.

i. Wählen Sie Aktionen und dann Hauptaktualisierung aus.

ii. Füllen Sie das Formular aus und wählen Sie Weiter.

iii. Füllen Sie die Formulare aus und wählen Sie Weiter, bis Sie zur Seite „Überprüfen“ gelangen.

iv. Überprüfen Sie Ihre Aktualisierungen und wählen Sie Aktualisieren.

AWS CLI

Verwenden Sie den AWS Proton AWS CLI , um eine Umgebung auf eine neue Nebenversion zu aktualisieren.

Führen Sie den folgenden Befehl aus, um Ihre Umgebung zu aktualisieren:

```
$ 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"
```

Antwort:

```
{
    "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"
    }
}
```

Führen Sie den folgenden Befehl aus, um den Status abzurufen und zu bestätigen:

```
$ aws proton get-environment \
        --name "MySimpleEnv"
```

Antwort:

```
{
    "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"
    }
}
```

Aktualisieren Sie eine selbstverwaltete Bereitstellungsumgebung

Selbstverwaltete Bereitstellung wird nur von Umgebungen unterstützt, die mit Terraform bereitstellen.

Verwenden Sie die Konsole oder AWS CLI aktualisieren Sie Ihre Umgebung.

AWS-Managementkonsole

Aktualisieren Sie eine Umgebung mithilfe der Konsole, wie in den folgenden Schritten gezeigt.

1. Wählen Sie einen der folgenden 2 Schritte aus.
   a. In der Liste der Umgebungen.
      i. Wählen Sie in der AWS Proton Konsole Umgebungen aus.
      ii. Wählen Sie in der Liste der Umgebungen das Optionsfeld links neben der Umgebungsvorlage aus, die Sie aktualisieren möchten.
   b. Auf der Detailseite der Konsolenumgebung.
      i. Wählen Sie in der AWS Proton Konsole Umgebungen aus.
      ii. Wählen Sie in der Liste der Umgebungen den Namen der Umgebung aus, die Sie aktualisieren möchten.
2. Wählen Sie einen der nächsten 4 Schritte aus, um Ihre Umgebung zu aktualisieren.
   a. Um eine Änderung vorzunehmen, für die keine Bereitstellung der Umgebung erforderlich ist.
      i. Zum Beispiel, um eine Beschreibung zu ändern.

         Wählen Sie Bearbeiten aus.
      ii. Füllen Sie das Formular aus und wählen Sie Weiter.
      iii. Überprüfe deine Bearbeitung und wähle „Aktualisieren“.
   b. Um nur die Metadateneingaben zu aktualisieren.
      i. Wählen Sie Aktionen und dann Aktualisieren aus.
      ii. Füllen Sie das Formular aus und wählen Sie Bearbeiten.
      iii. Füllen Sie die Formulare aus und wählen Sie Weiter, bis Sie zur Seite „Überprüfen“ gelangen.
      iv. Überprüfen Sie Ihre Aktualisierungen und wählen Sie Aktualisieren.
   c. Um eine neue Nebenversion der zugehörigen Umgebungsvorlage zu aktualisieren.
      i. Wählen Sie „Aktionen“ und dann „Nebenprogramm aktualisieren“.
      ii. Füllen Sie das Formular aus und wählen Sie Weiter.

iii. Füllen Sie die Formulare aus und wählen Sie Weiter, bis Sie zur Seite „Überprüfen“ gelangen.

iv. Überprüfen Sie Ihre Aktualisierungen und wählen Sie Aktualisieren.

d. Um eine neue Hauptversion der zugehörigen Umgebungsvorlage zu aktualisieren.

i. Wählen Sie Aktionen und dann Hauptaktualisierung aus.

ii. Füllen Sie das Formular aus und wählen Sie Weiter.

iii. Füllen Sie die Formulare aus und wählen Sie Weiter, bis Sie zur Seite „Überprüfen“ gelangen.

iv. Überprüfen Sie Ihre Aktualisierungen und wählen Sie Aktualisieren.

AWS CLI

Verwenden Sie die AWS CLI , um eine Terraform-Umgebung auf eine neue Nebenversion mit selbstverwalteter Bereitstellung zu aktualisieren.

1. Führen Sie den folgenden Befehl aus, um Ihre Umgebung zu aktualisieren:

```
$ 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"
```

Antwort:

```
{
    "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. Führen Sie den folgenden Befehl aus, um den Status abzurufen und zu bestätigen:

```
$ aws proton get-environment \
    --name "pr-environment"
```

Antwort:

```
{
    "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. Überprüfen Sie die Pull-Anfrage, die von gesendet wurde AWS Proton.

   - Wenn Sie die Anfrage genehmigen, ist die Bereitstellung im Gange.
   - Wenn Sie die Anfrage ablehnen, wird die Erstellung der Umgebung abgebrochen.
   - Wenn bei der Pull-Anfrage ein Timeout auftritt, ist die Erstellung der Umgebung nicht abgeschlossen.

4. Geben Sie den Bereitstellungsstatus an AWS Proton.

```
$ aws proton notify-resource-deployment-status-change \
    --resource-arn "arn:aws:proton:region-id:123456789012:environment/pr-environment" \
    --status "SUCCEEDED"
```

Brechen Sie eine laufende Umgebungsbereitstellung ab

Sie können versuchen, die Bereitstellung eines Umgebungsupdates abzubrechen, wenn das aktiviert `deploymentStatus` ist`IN_PROGRESS`. AWS Proton versucht, die Bereitstellung abzubrechen. Eine erfolgreiche Stornierung ist nicht garantiert.

Wenn Sie eine Update-Bereitstellung stornieren, wird AWS Proton versucht, die Bereitstellung wie in den folgenden Schritten beschrieben abzubrechen.

AWS Proton Führt mit AWS-managed Provisioning Folgendes aus:

- Setzt den Bereitstellungsstatus auf. `CANCELLING`
- Stoppt die laufende Bereitstellung und löscht alle neuen Ressourcen, die durch die Bereitstellung erstellt wurden, wann`IN_PROGRESS`.
- Setzt den Bereitstellungsstatus auf`CANCELLED`.
- Setzt den Zustand der Ressource auf den Zustand vor dem Start der Bereitstellung zurück.

AWS Proton Führt bei der selbstverwalteten Bereitstellung Folgendes aus:

- Versucht, die Pull-Anfrage zu schließen, um zu verhindern, dass die Änderungen an Ihrem Repository zusammengeführt werden.
- Setzt den Bereitstellungsstatus auf, `CANCELLED` wenn die Pull-Anfrage erfolgreich geschlossen wurde.

Anweisungen zum Abbrechen einer Umgebungsbereitstellung finden Sie CancelEnvironmentDeploymentin der AWS Proton API-Referenz.

Sie können die Konsole oder CLI verwenden, um laufende Umgebungen abzubrechen.

AWS-Managementkonsole

Verwenden Sie die Konsole, um die Bereitstellung eines Umgebungsupdates abzubrechen, wie in den folgenden Schritten gezeigt.

1. Wählen Sie in der AWS Proton Konsole im Navigationsbereich Umgebungen aus.
2. Wählen Sie in der Liste der Umgebungen den Namen der Umgebung mit dem Bereitstellungsupdate aus, das Sie stornieren möchten.
3. Wenn der Status der Update-Bereitstellung auf In Bearbeitung steht, wählen Sie auf der Seite mit den Umgebungsdetails die Option Aktionen und dann Bereitstellung abbrechen aus.
4. In einem Fenster werden Sie aufgefordert, zu bestätigen, dass Sie den Vorgang abbrechen möchten. Wählen Sie Bereitstellung abbrechen aus.
5. Ihr Status für die Bereitstellung von Updates ist auf Storniert und dann auf Storniert gesetzt, um die Stornierung abzuschließen.

AWS CLI

Verwenden Sie den AWS Proton AWS CLI , um die Bereitstellung eines IN_PROGRESS-Umgebungsupdates auf eine neue Nebenversion abzubrechen 2.

In der für dieses Beispiel verwendeten Vorlage ist eine Wartebedingung enthalten, sodass der Abbruch beginnt, bevor das Update erfolgreich bereitgestellt wird.

Führen Sie den folgenden Befehl aus, um das Update abzubrechen:

```
$ aws proton cancel-environment-deployment \
        --environment-name "MySimpleEnv"
```

Antwort:

```
{
    "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"
    }
}
```

Führen Sie den folgenden Befehl aus, um den Status abzurufen und zu bestätigen:“

```
$ aws proton get-environment \
        --name "MySimpleEnv"
```

Antwort:

```
{
    "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"
    }
}
```

Löschen Sie eine Umgebung

Sie können eine AWS Proton Umgebung mithilfe der AWS Proton Konsole oder der löschen AWS CLI.

> **Note**
>
> Sie können keine Umgebung löschen, der eine Komponente zugeordnet ist. Um eine solche Umgebung zu löschen, sollten Sie zunächst alle Komponenten löschen, die in der Umgebung ausgeführt werden. Weitere Informationen zu Komponenten finden Sie unterKomponenten.

AWS-Managementkonsole

Löschen Sie eine Umgebung mithilfe der Konsole, wie in den folgenden beiden Optionen beschrieben.

In der Liste der Umgebungen.

1. Wählen Sie in der AWS Proton Konsole Umgebungen aus.
2. Wählen Sie in der Liste der Umgebungen das Optionsfeld links neben der Umgebung aus, die Sie löschen möchten.
3. Wählen Sie Actions (Aktionen) und anschließend Delete (Löschen) aus.
4. In einem Modal werden Sie aufgefordert, die Löschaktion zu bestätigen.
5. Folgen Sie den Anweisungen und wählen Sie Ja, löschen.

Auf der Seite mit den Umgebungsdetails.

1. Wählen Sie in der AWS Proton Konsole Umgebungen aus.
2. Wählen Sie in der Liste der Umgebungen den Namen der Umgebung aus, die Sie löschen möchten.
3. Wählen Sie auf der Detailseite der Umgebung Aktionen und dann Löschen aus.
4. In einem Modalfenster werden Sie aufgefordert, zu bestätigen, dass Sie löschen möchten.
5. Folgen Sie den Anweisungen und wählen Sie Ja, löschen.

AWS CLI

Verwenden Sie die AWS CLI , um eine Umgebung zu löschen.

Löschen Sie eine Umgebung nicht, wenn Dienste oder Dienstinstanzen in der Umgebung bereitgestellt werden.

Führen Sie den folgenden Befehl aus:

```
$ aws proton delete-environment \
    --name "MySimpleEnv"
```

Antwort:

```
{
    "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"
    }
}
```

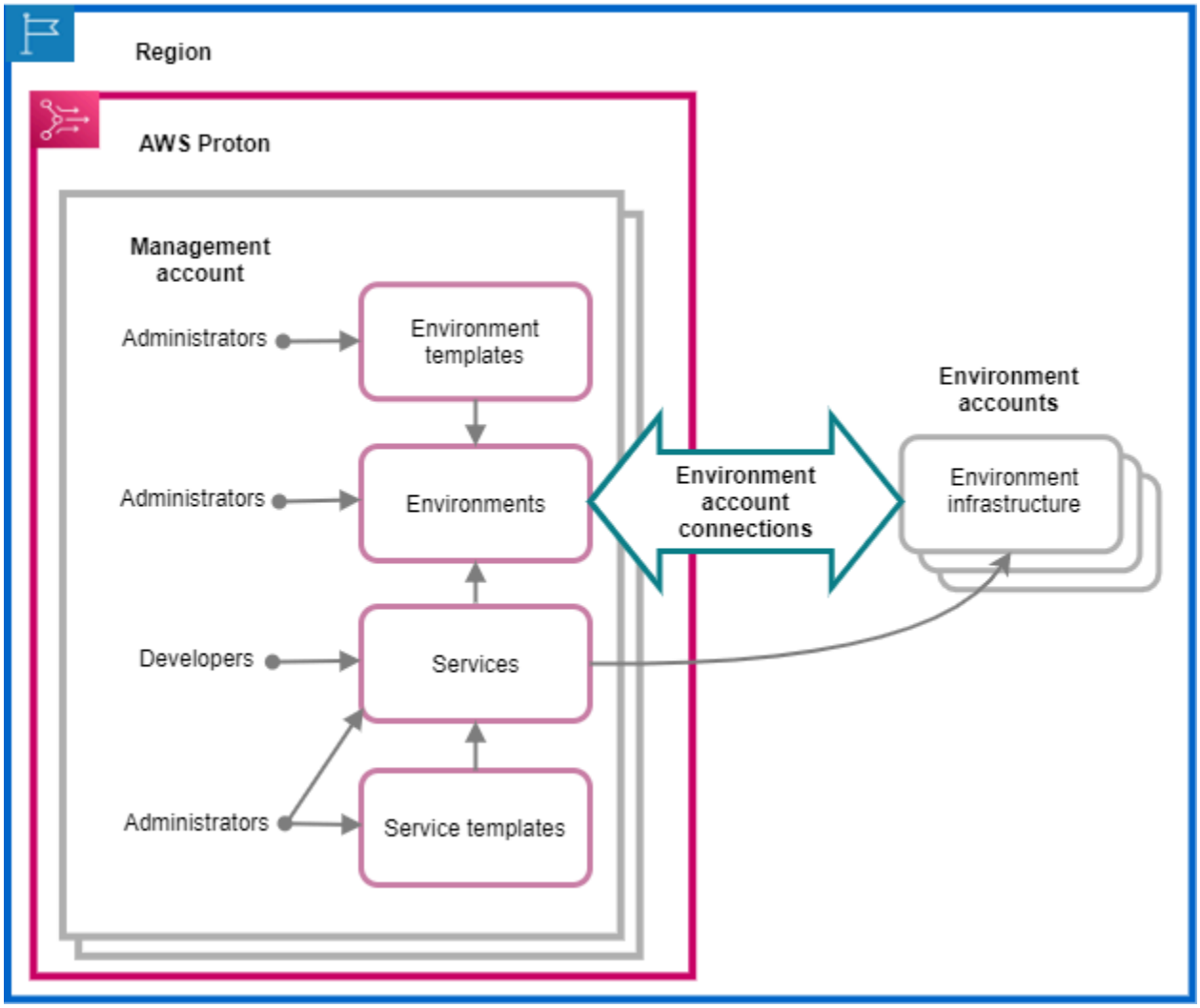
Verbindungen zu Umgebungskonten

Übersicht

Erfahren Sie, wie Sie eine AWS Proton Umgebung in einem Konto erstellen und verwalten und deren Infrastrukturressourcen in einem anderen Konto bereitstellen. Dies kann dazu beitragen, die Transparenz und Effizienz im großen Maßstab zu verbessern. Verbindungen mit Umgebungskonten unterstützen nur die Standardbereitstellung mit CloudFormation Infrastruktur als Code.

> **Note**
>
> Die Informationen in diesem Thema sind für Umgebungen relevant, die mit AWS verwalteter Bereitstellung konfiguriert sind. Bei Umgebungen, die mit selbstverwalteter Bereitstellung konfiguriert sind, wird AWS Proton Ihre Infrastruktur nicht direkt bereitgestellt. Stattdessen werden Pull-Requests (PRs) zur Bereitstellung an Ihr Repository gesendet. Es liegt in Ihrer Verantwortung, sicherzustellen, dass Ihr Automatisierungscode die richtige Identität und Rolle annimmt.
> Weitere Informationen zu Bereitstellungsmethoden finden Sie unterthe section called “Bereitstellungsmethoden”.

Terminologie



Mit Verbindungen zu AWS Proton Umgebungskonten können Sie aus einem Konto eine AWS Proton Umgebung erstellen und deren Infrastruktur in einem anderen Konto bereitstellen.

Verwaltungskonto

Das einzige Konto, mit dem Sie als Administrator eine AWS Proton Umgebung erstellen, die Infrastrukturressourcen in einem anderen Umgebungskonto bereitstellt.

Umgebungskonto

Ein Konto, in dem die Umgebungsinfrastruktur bereitgestellt wird, wenn Sie eine AWS Proton Umgebung in einem anderen Konto erstellen.

Verbindung zum Umgebungskonto

Eine sichere bidirektionale Verbindung zwischen einem Verwaltungskonto und einem Umgebungskonto. Es behält die Autorisierung und die Berechtigungen bei, wie in den folgenden Abschnitten näher beschrieben.

Wenn Sie eine Umgebungskontoverbindung in einem Umgebungskonto in einer bestimmten Region erstellen, können nur die Verwaltungskonten in derselben Region die Umgebungskontoverbindung sehen und verwenden. Das bedeutet, dass sich die im Verwaltungskonto erstellte AWS Proton Umgebung und die im Umgebungskonto bereitgestellte Umgebungsinfrastruktur in derselben Region befinden müssen.

Überlegungen zur Verbindung mit dem Umgebungskonto

- Sie benötigen für jede Umgebung, die Sie in einem Umgebungskonto bereitstellen möchten, eine Verbindung mit einem Umgebungskonto.
- Informationen zu Verbindungskontingenten für Umgebungskonten finden Sie unterAWS Proton Kontingente.

Tagging

Verwenden Sie im Umgebungskonto die Konsole oder die vom Kunden verwalteten Tags AWS CLI , um die Verbindung mit dem Umgebungskonto anzuzeigen und zu verwalten. AWS verwaltete Tags werden nicht für Verbindungen mit Umgebungskonten generiert. Weitere Informationen finden Sie unter Tagging.

Erstellen Sie eine Umgebung in einem Konto und stellen Sie deren Infrastruktur in einem anderen Konto bereit

Um eine Umgebung von einem einzigen Verwaltungskonto aus zu erstellen und bereitzustellen, richten Sie ein Umgebungskonto für eine Umgebung ein, die Sie erstellen möchten.

Starten Sie mit dem Umgebungskonto und stellen Sie eine Verbindung her.

Erstellen Sie im Umgebungskonto eine AWS Proton Servicerolle, die nur auf die Berechtigungen beschränkt ist, die für die Bereitstellung der Infrastrukturressourcen Ihrer Umgebung erforderlich sind. Weitere Informationen finden Sie unter AWS Proton Servicerolle für die Bereitstellung mit CloudFormation.

Erstellen Sie anschließend eine Verbindungsanfrage für das Umgebungskonto und senden Sie sie an Ihr Verwaltungskonto. Wenn die Anfrage akzeptiert wurde, AWS Proton kann die zugehörige IAM-Rolle verwendet werden, die die Bereitstellung von Umgebungsressourcen im zugehörigen Umgebungskonto ermöglicht.

Nehmen Sie im Verwaltungskonto die Verbindung mit dem Umgebungskonto an oder lehnen Sie sie ab.

Nehmen Sie im Verwaltungskonto die Verbindungsanfrage für das Umgebungskonto an oder lehnen Sie sie ab. Sie können eine Verbindung mit einem Umgebungskonto nicht aus Ihrem Verwaltungskonto löschen.

Wenn Sie die Anfrage akzeptieren, AWS Proton können sie die zugehörige IAM-Rolle verwenden, die die Bereitstellung von Ressourcen im zugehörigen Umgebungskonto ermöglicht.

Die Ressourcen der Umgebungsinfrastruktur werden im zugehörigen Umgebungskonto bereitgestellt. Sie können AWS Proton APIs nur von Ihrem Verwaltungskonto aus auf Ihre Umgebung und deren Infrastrukturressourcen zugreifen und diese verwalten. Weitere Informationen erhalten Sie unter Erstellen Sie eine Umgebung in einem Konto und stellen Sie sie in einem anderen Konto bereit und Aktualisieren einer Umgebung.

Nachdem Sie eine Anfrage abgelehnt haben, können Sie die abgelehnte Verbindung mit dem Umgebungskonto weder annehmen noch verwenden.

> **Note**
>
> Sie können eine Verbindung mit einem Umgebungskonto, die mit einer Umgebung verbunden ist, nicht ablehnen. Um die Verbindung mit dem Umgebungskonto abzulehnen, müssen Sie zuerst die zugehörige Umgebung löschen.

Greifen Sie im Umgebungskonto auf die bereitgestellten Infrastrukturressourcen zu.

Im Umgebungskonto können Sie die bereitgestellten Infrastrukturressourcen anzeigen und darauf zugreifen. Sie können beispielsweise CloudFormation API-Aktionen verwenden, um Stacks zu überwachen und bei Bedarf zu bereinigen. Sie können die AWS Proton API-Aktionen nicht verwenden, um auf die AWS Proton Umgebung zuzugreifen oder diese zu verwalten, die für die Bereitstellung der Infrastrukturressourcen verwendet wurde.

Im Umgebungskonto können Sie Verbindungen mit Umgebungskonten löschen, die Sie im Umgebungskonto erstellt haben. Sie können sie nicht akzeptieren oder ablehnen. Wenn Sie eine Umgebungskontoverbindung löschen, die von einer AWS Proton Umgebung verwendet AWS Proton wird, können die Ressourcen der Umgebungsinfrastruktur erst verwaltet werden, wenn eine neue Umgebungsverbindung für das Umgebungskonto und die benannte Umgebung akzeptiert wurde. Sie sind dafür verantwortlich, bereitgestellte Ressourcen zu bereinigen, die noch keine Verbindung zur Umgebung haben.

Verwenden Sie die Konsole oder CLI, um die Verbindungen zu Umgebungskonten zu verwalten

Sie können die Konsole oder CLI verwenden, um Verbindungen zu Umgebungskonten zu erstellen und zu verwalten.

AWS-Managementkonsole

Verwenden Sie die Konsole, um eine Verbindung mit einem Umgebungskonto herzustellen und eine Anfrage an das Verwaltungskonto zu senden, wie in den nächsten Schritten gezeigt.

1. Entscheiden Sie sich für einen Namen für die Umgebung, die Sie in Ihrem Verwaltungskonto erstellen möchten, oder wählen Sie den Namen einer vorhandenen Umgebung, für die eine Verbindung mit einem Umgebungskonto erforderlich ist.
2. Wählen Sie in einem Umgebungskonto in der AWS Proton Konsole im Navigationsbereich die Option Environment account connections aus.

3. Wählen Sie auf der Seite Verbindungen zu Umgebungskonten die Option Verbindungsanfrage aus.

> **Note**
>
> Überprüfen Sie die Konto-ID, die in der Überschrift der Seite „Verbindung mit Umweltkonten“ aufgeführt ist. Stellen Sie sicher, dass sie mit der Konto-ID des Umgebungskontos übereinstimmt, in dem Ihre benannte Umgebung bereitgestellt werden soll.

4. Gehen Sie auf der Seite Verbindungsanfrage wie folgt vor:

   a. Geben Sie im Abschnitt Mit Verwaltungskonto verbinden die Verwaltungskonto-ID und den Umgebungsnamen ein, die Sie in Schritt 1 eingegeben haben.

   b. Wählen Sie im Abschnitt Umgebungsrolle die Option Neue Servicerolle aus. AWS Proton Daraufhin wird automatisch eine neue Rolle für Sie erstellt. Oder wählen Sie Bestehende Servicerolle und den Namen der Servicerolle aus, die Sie zuvor erstellt haben.

> **Note**
>
> Die Rolle, die AWS Proton automatisch für Sie erstellt wird, verfügt über umfassende Berechtigungen. Es wird empfohlen, die Rolle auf die Berechtigungen zu beschränken, die für die Bereitstellung der Infrastrukturressourcen Ihrer Umgebung erforderlich sind. Weitere Informationen finden Sie unter AWS Proton Servicerolle für die Bereitstellung mit CloudFormation.

   c. (Optional) Wählen Sie im Abschnitt Tags die Option Neues Tag hinzufügen aus, um ein vom Kunden verwaltetes Tag für Ihre Umgebungskontoverbindung zu erstellen.

   d. Wählen Sie Verbindungsanfrage aus.

5. Ihre Anfrage wird in der Umgebung als ausstehend angezeigt. Verbindungen, die an eine Verwaltungskontotabelle gesendet wurden, und in einem Modalfenster erfahren Sie, wie Sie die Anfrage vom Verwaltungskonto annehmen können.

Nehmen Sie eine Verbindungsanfrage für ein Umgebungskonto an oder lehnen Sie sie ab.

1. Wählen Sie in einem Verwaltungskonto in der AWS Proton Konsole im Navigationsbereich die Option Environment account connections aus.
2. Wählen Sie auf der Seite Verbindungen zu Umgebungskonten in der Tabelle Verbindungsanfragen für Umgebungskonten die Verbindungsanfrage für die Umgebung aus, die Sie annehmen oder ablehnen möchten.

   > **Note**
   >
   > Überprüfen Sie die Konto-ID, die in der Überschrift der Seite „Verbindung mit Umweltkonten“ aufgeführt ist. Stellen Sie sicher, dass sie mit der Konto-ID des Verwaltungskontos übereinstimmt, das mit der Verbindung zum Umgebungskonto verknüpft ist, die abgelehnt werden soll. Nachdem Sie diese Verbindung mit dem Umgebungskonto abgelehnt haben, können Sie die abgelehnte Verbindung mit dem Umgebungskonto weder akzeptieren noch verwenden.

3. Wählen Sie Ablehnen oder Akzeptieren.

   - Wenn Sie Ablehnen ausgewählt haben, ändert sich der Status von „Ausstehend“ in „Abgelehnt“.
   - Wenn Sie „Annehmen“ ausgewählt haben, ändert sich der Status von „Ausstehend“ in „Verbunden“.

Löscht eine Verbindung mit einem Umgebungskonto.

1. Wählen Sie in einem Umgebungskonto in der AWS Proton Konsole im Navigationsbereich die Option Environment account connections aus.

   > **Note**
   >
   > Überprüfen Sie die Konto-ID, die in der Überschrift der Seite „Verbindung mit Umweltkonten“ aufgeführt ist. Stellen Sie sicher, dass sie mit der Konto-ID des Verwaltungskontos übereinstimmt, das mit der Verbindung zum Umgebungskonto verknüpft ist, die abgelehnt werden soll. Nachdem Sie diese Verbindung mit dem Umgebungskonto gelöscht haben, AWS Proton können die Ressourcen der Umgebungsinfrastruktur im Umgebungskonto nicht verwaltet werden. Es kann erst verwaltet werden, nachdem eine neue Verbindung mit einem Umgebungskonto für

das Umgebungskonto und die benannte Umgebung vom Verwaltungskonto akzeptiert wurde.

2. Wählen Sie auf der Seite Verbindungen mit Umgebungskonten im Abschnitt Gesendete Anfragen zur Verbindung mit dem Verwaltungskonto die Option Löschen aus.
3. In einem Modalfenster werden Sie aufgefordert, zu bestätigen, dass Sie den Vorgang löschen möchten. Wählen Sie Löschen aus.

AWS CLI

Entscheiden Sie sich für einen Namen für die Umgebung, die Sie in Ihrem Verwaltungskonto erstellen möchten, oder wählen Sie den Namen einer vorhandenen Umgebung, für die eine Verbindung mit einem Umgebungskonto erforderlich ist.

Erstellen Sie eine Verbindung mit einem Umgebungskonto in einem Umgebungskonto.

Führen Sie den folgenden Befehl aus:

```
$ 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"
```

Antwort:

```
{
    "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"
    }
```

```
}
```

Akzeptieren oder lehnen Sie eine Verbindung mit einem Umgebungskonto in einem Verwaltungskonto ab, wie im folgenden Befehl und der folgenden Antwort gezeigt.

> **Note**
>
> Wenn Sie diese Verbindung mit dem Umgebungskonto ablehnen, können Sie die abgelehnte Verbindung mit dem Umgebungskonto nicht akzeptieren oder verwenden.

Wenn Sie Ablehnen angeben, ändert sich der Status von „Ausstehend“ in „Abgelehnt“.

Wenn Sie Akzeptieren angeben, ändert sich der Status von „Ausstehend“ in „Verbunden“.

Führen Sie den folgenden Befehl aus, um die Verbindung mit dem Umgebungskonto zu akzeptieren:

```
$ aws proton accept-environment-account-connection \
    --id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

Antwort:

```
{
    "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"
    }
}
```

Führen Sie den folgenden Befehl aus, um die Verbindung mit dem Umgebungskonto abzulehnen:

```
$ aws proton reject-environment-account-connection \
    --id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

Antwort:

```
{
    "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"
    }
}
```

Sehen Sie sich die Verbindungen eines Umgebungskontos an. Sie können Verbindungen zu Umgebungskonten abrufen oder auflisten.

Führen Sie den folgenden Befehl get aus:

```
$ aws proton get-environment-account-connection \
    --id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

Antwort:

```
{
    "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"
    }
}
```

Löschen Sie eine Umgebungskontoverbindung in einem Umgebungskonto.

> **Note**
>
> Wenn Sie diese Umgebungskontoverbindung löschen, AWS Proton können Sie die Umgebungsinfrastrukturressourcen im Umgebungskonto erst verwalten, wenn eine neue Umgebungsverbindung für das Umgebungskonto und die benannte Umgebung akzeptiert wurde. Sie sind dafür verantwortlich, bereitgestellte Ressourcen zu bereinigen, die noch keine Verbindung zur Umgebung haben.

Führen Sie den folgenden Befehl aus:

```
$ aws proton delete-environment-account-connection \
    --id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

Antwort:

```
{
    "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"
    }
}
```

Vom Kunden verwaltete Umgebungen

In kundenverwalteten Umgebungen können Sie die bestehende Infrastruktur wie eine VPC verwenden, die Sie bereits als Ihre AWS Proton Umgebung bereitgestellt haben. Wenn Sie vom Kunden verwaltete Umgebungen verwenden, können Sie Ihre eigenen gemeinsam genutzten Ressourcen außerhalb von bereitstellen. AWS Proton Sie können jedoch AWS Proton weiterhin zulassen, dass relevante Bereitstellungsausgaben als Eingaben für AWS Proton Dienste verwendet werden, wenn diese bereitgestellt werden. Wenn sich die Ausgaben ändern können, AWS Proton ist in der Lage, Aktualisierungen zu akzeptieren. AWS Proton ist jedoch nicht in der Lage, die Umgebung direkt zu ändern, da die Bereitstellung außerhalb von AWS Proton verwaltet wird.

Nachdem die Umgebung erstellt wurde, sind Sie dafür verantwortlich, dieselben Ausgaben bereitzustellen AWS Proton , die auch bei der Erstellung der Umgebung erstellt worden wären, z. B. Amazon ECS-Clusternamen oder Amazon VPC IDs. AWS Proton

Mit dieser Funktion können Sie Serviceressourcen aus einer AWS Proton AWS Proton Service-Vorlage in dieser Umgebung bereitstellen und aktualisieren. Die Umgebung selbst wird jedoch nicht durch Vorlagenaktualisierungen in geändert AWS Proton. Sie sind dafür verantwortlich, Aktualisierungen an der Umgebung durchzuführen und diese Ausgaben in zu aktualisieren AWS Proton.

Sie können mehrere Umgebungen in einem einzigen Konto haben, bei denen es sich um eine Mischung aus AWS Proton verwalteten und kundenverwalteten Umgebungen handelt. Sie können auch ein zweites Konto verknüpfen und eine AWS Proton Vorlage im primären Konto verwenden, um Bereitstellungen und Updates für Umgebungen und Dienste in diesem zweiten, verknüpften Konto auszuführen.

Wie verwendet man vom Kunden verwaltete Umgebungen

Als Erstes müssen Administratoren eine importierte, vom Kunden verwaltete Umgebungsvorlage registrieren. Geben Sie im Vorlagenpaket keine Manifeste oder Infrastrukturdateien an. Geben Sie nur das Schema an.

Das folgende Schema skizziert eine Liste von Ausgaben, die das offene API-Format verwenden, und repliziert die Ausgaben aus einer CloudFormation Vorlage.

> **Important**
>
> Für die Ausgaben sind nur Zeichenketteneingaben zulässig.

Das folgende Beispiel ist ein Ausschnitt aus den Ausgabeabschnitten einer CloudFormation Vorlage für eine entsprechende Fargate-Vorlage.

```
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'
[...]
```

Das Schema für die entsprechende AWS Proton importierte Umgebung ähnelt dem Folgenden. Geben Sie im Schema keine Standardwerte an.

```
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"
[...]
```

Bei der Registrierung der Vorlage geben Sie an, dass diese Vorlage importiert wurde und den Amazon S3 S3-Bucket-Speicherort für das Paket bereitstellt. AWS Proton überprüft, ob das Schema nur Vorlagenparameter `environment_input_type` und keine CloudFormation Vorlagenparameter enthält, bevor die Vorlage in den Entwurf aufgenommen wird.

Sie geben Folgendes an, um eine importierte Umgebung zu erstellen.

- Eine IAM-Rolle, die bei Bereitstellungen verwendet werden soll.
- Eine Spezifikation mit den Werten für die erforderlichen Ausgaben.

Sie können beide Optionen entweder über die Konsole oder AWS CLI mithilfe eines Verfahrens bereitstellen, das der Bereitstellung einer regulären Umgebung ähnelt.

CodeBuild Erstellung von Bereitstellungsrollen

IaaC-Tools (Infrastructure as a Code) wie Terraform CloudFormation und Terraform benötigen Berechtigungen für die vielen verschiedenen Arten von Ressourcen. AWS Wenn eine IaaC-Vorlage beispielsweise einen Amazon S3 S3-Bucket deklariert, benötigt sie Berechtigungen zum Erstellen, Lesen, Aktualisieren und Löschen von Amazon S3 S3-Buckets. Es wird als bewährte Sicherheitsmethode angesehen, Rollen auf die minimal erforderlichen Berechtigungen zu beschränken. Angesichts der Vielzahl der AWS Ressourcen ist es schwierig, Richtlinien mit den geringsten Rechten für IaaC-Vorlagen zu erstellen, insbesondere wenn sich die Ressourcen, die von diesen Vorlagen verwaltet werden, später ändern können. Beispielsweise fügen Sie bei Ihren letzten Änderungen an einer Vorlage, von der verwaltet wird AWS Proton, eine RDS-Datenbankressource hinzu.

Die Konfiguration der richtigen Berechtigungen trägt zu einer reibungslosen Bereitstellung Ihres IaC bei. AWS Proton CodeBuild Beim Provisioning werden beliebige vom Kunden bereitgestellte CLI-Befehle in einem CodeBuild Projekt ausgeführt, das sich im Konto des Kunden befindet. In der Regel erstellen und löschen diese Befehle die Infrastruktur mithilfe eines IaaC-Tools (Infrastructure as Code) wie. AWS CDK Wenn eine AWS Ressource bereitgestellt wird, deren Vorlage CodeBuild Provisioning verwendet, AWS wird ein Build in einem CodeBuild Projekt gestartet, das von verwaltet wird. AWS Es wird eine Rolle an übergeben CodeBuild, die davon CodeBuild ausgeht, Befehle auszuführen. Diese Rolle, die sogenannte CodeBuild Bereitstellungsrolle, wird vom Kunden bereitgestellt und enthält die für die Bereitstellung der Infrastruktur erforderlichen Berechtigungen. Sie darf nur von ihnen übernommen werden CodeBuild und AWS Proton kann sie auch nicht annehmen.

Erstellen der -Rolle

Die CodeBuild Provisioning-Rolle kann in der IAM-Konsole oder in der erstellt werden. AWS CLI Um sie zu erstellen in: 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
```

Damit wird auch der angehängt`AWSProtonCodeBuildProvisioningBasicAccess`, der die Mindestberechtigungen enthält, die der CodeBuild Dienst zum Ausführen eines Builds benötigt.

Wenn Sie lieber die Konsole verwenden möchten, stellen Sie bei der Erstellung der Rolle bitte Folgendes sicher:

1. Wählen Sie für vertrauenswürdige Entität AWS Dienst und dann aus CodeBuild.
2. Wählen Sie im Schritt Berechtigungen hinzufügen alle anderen Richtlinien aus`AWSProtonCodeBuildProvisioningBasicAccess`, die Sie anhängen möchten.

Administratorzugriff

Wenn Sie die `AdministratorAccess` Richtlinie an die CodeBuild Bereitstellungsrolle anhängen, wird garantiert, dass keine IaaC-Vorlage aufgrund fehlender Berechtigungen fehlschlägt. Das bedeutet auch, dass jeder, der eine Umgebungs- oder Dienstvorlage erstellen kann, Aktionen auf Administratorebene ausführen kann, auch wenn dieser Benutzer kein Administrator ist. AWS Proton empfiehlt die Verwendung `AdministatorAccess` zusammen mit der Provisioning-Rolle nicht. CodeBuild Wenn Sie sich für die Verwendung `AdministratorAccess` mit der CodeBuild Provisioning-Rolle entscheiden, tun Sie dies in einer Sandbox-Umgebung.

Sie können eine Rolle `AdministratorAccess` in der IAM-Konsole oder durch Ausführen dieses Befehls erstellen:

```
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
```

Eine Rolle mit minimalem Geltungsbereich erstellen

Wenn Sie eine Rolle mit Mindestberechtigungen erstellen möchten, gibt es mehrere Möglichkeiten:

- Stellen Sie die Lösung mit Administratorberechtigungen bereit und beschränken Sie dann den Umfang der Rolle. Wir empfehlen die Verwendung von IAM Access Analyzer.
- Verwenden Sie verwaltete Richtlinien, um Zugriff auf die Dienste zu gewähren, die Sie nutzen möchten.

AWS CDK

Wenn Sie AWS CDK mit AWS Proton verwenden und in jedem Umgebungskonto/jeder Region `cdk bootstrap` ausgeführt haben, gibt es bereits eine Rolle für. `cdk deploy` Fügen Sie in diesem Fall der CodeBuild Bereitstellungsrolle die folgende Richtlinie hinzu:

```
{
    "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"
}
```

Benutzerdefiniertes VPC

Wenn Sie sich für die Ausführung CodeBuild in einer benutzerdefinierten VPC entscheiden, benötigen Sie in Ihrer CodeBuild Rolle die folgenden Berechtigungen:

```
{
    "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"
        }
    }
}
```

Sie könnten auch die AmazonEC2FullAccess verwaltete Richtlinie verwenden, obwohl diese auch Berechtigungen beinhaltet, die Sie möglicherweise nicht benötigen. So hängen Sie die verwaltete Richtlinie mit der CLI an:

```
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 Dienstleistungen

Ein AWS Proton Service ist eine Instanziierung einer Dienstvorlage, die normalerweise mehrere Dienstinstanzen und eine Pipeline umfasst. Eine AWS Proton Dienstinstanz ist eine Instanziierung einer Dienstvorlage in einer bestimmten Umgebung. Eine Dienstvorlage ist eine vollständige Definition der Infrastruktur und der optionalen Servicepipeline für einen AWS Proton Dienst.

Nachdem Sie Ihre Dienstinstanzen bereitgestellt haben, können Sie sie aktualisieren, indem Sie Quellcode-Pushs eingeben, die die CI/CD Pipeline dazu auffordern, oder indem Sie den Service auf neue Versionen seiner Dienstvorlage aktualisieren. AWS Proton fordert Sie auf, wenn neue Versionen der Dienstvorlage verfügbar sind, sodass Sie Ihre Dienste auf eine neue Version aktualisieren können. Wenn Ihr Service aktualisiert wird, werden der Service und die AWS Proton Dienstinstanzen erneut bereitgestellt.

In diesem Kapitel wird gezeigt, wie Sie Dienste mithilfe von Erstellungs-, Anzeige-, Aktualisierungs- und Löschvorgängen verwalten. Weitere Informationen finden Sie in der AWS Proton Service-API-Referenz.

Themen

Einen Service erstellen

Um eine Anwendung bereitzustellen AWS Proton, erstellen Sie als Entwickler einen Dienst und geben die folgenden Eingaben ein.

1. Der Name einer AWS Proton Dienstvorlage, die vom Plattformteam veröffentlicht wurde.
2. Ein Name für den Dienst.

3. Die Anzahl der Dienstinstanzen, die Sie bereitstellen möchten.
4. Eine Auswahl von Umgebungen, die Sie verwenden möchten.
5. Eine Verbindung zu Ihrem Code-Repository, wenn Sie eine Dienstvorlage verwenden, die eine Service-Pipeline enthält (optional).

Was ist in einem Service enthalten?

Wenn Sie einen AWS Proton Service erstellen, können Sie aus zwei verschiedenen Typen von Dienstvorlagen wählen:

- Eine Dienstvorlage, die eine Service-Pipeline enthält (Standard).
- Eine Dienstvorlage, die keine Servicepipeline enthält.

Sie müssen mindestens eine Dienstinstanz erstellen, wenn Sie Ihren Service erstellen.

Eine Dienstinstanz und eine optionale Pipeline sind einem Dienst zugeordnet. Sie können eine Pipeline nur im Rahmen der Aktionen zum Erstellen und Löschen von Diensten erstellen oder löschen. Informationen zum Hinzufügen und Entfernen von Instanzen zu einem Service finden Sie unterBearbeiten Sie einen Dienst.

> **Note**
>
> Ihre Umgebung ist entweder für AWS— oder für selbstverwaltetes Provisioning konfiguriert. AWS Proton stellt Dienste in einer Umgebung bereit und verwendet dabei dieselbe Bereitstellungsmethode wie die Umgebung. Der Entwickler, der Dienstinstanzen erstellt oder aktualisiert, sieht den Unterschied nicht und seine Erfahrung ist in beiden Fällen identisch. Weitere Informationen zu Bereitstellungsmethoden finden Sie unterthe section called “Bereitstellungsmethoden”.

Vorlagen für Dienste

Es sind sowohl Haupt- als auch Nebenversionen von Dienstvorlagen verfügbar. Wenn Sie die Konsole verwenden, wählen Sie die neueste `Recommended` Haupt- und Nebenversion der Dienstvorlage aus. Wenn Sie die verwenden AWS CLI und nur die Hauptversion der Dienstvorlage angeben, geben Sie implizit deren neueste `Recommended` Nebenversion an.

Im Folgenden werden der Unterschied zwischen Haupt- und Nebenversionen der Vorlage und deren Verwendung beschrieben.

- Neue Versionen einer Vorlage werden `Recommended` veröffentlicht, sobald sie von einem Mitglied des Plattformteams genehmigt wurden. Das bedeutet, dass neue Dienste mit dieser Version erstellt werden und Sie aufgefordert werden, bestehende Dienste auf die neue Version zu aktualisieren.
- Dadurch AWS Proton kann das Plattformteam Serviceinstanzen automatisch auf eine neue Nebenversion einer Servicevorlage aktualisieren. Nebenversionen müssen abwärtskompatibel sein.
- Da bei Hauptversionen im Rahmen des Aktualisierungsvorgangs neue Eingaben erforderlich sind, müssen Sie Ihren Service auf eine Hauptversion seiner Dienstvorlage aktualisieren. Hauptversionen sind nicht abwärtskompatibel.

Einen Service erstellen

Die folgenden Verfahren zeigen, wie Sie die AWS Proton Konsole verwenden oder AWS CLI einen Dienst mit oder ohne Dienstpipeline erstellen.

AWS-Managementkonsole

Erstellen Sie einen Dienst, wie in den folgenden Konsolenschritten gezeigt.

1. Wählen Sie in der AWS Proton Konsole Dienste aus.
2. Wählen Sie Create service.
3. Wählen Sie auf der Seite „Dienstvorlage auswählen“ eine Vorlage aus und klicken Sie auf Konfigurieren.

   Wenn Sie keine aktivierte Pipeline verwenden möchten, wählen Sie für Ihren Service eine Vorlage aus, die mit Pipeline ausschließt gekennzeichnet ist.
4. Geben Sie auf der Seite Dienst konfigurieren im Abschnitt Diensteinstellungen einen Dienstnamen ein.
5. (Optional) Geben Sie eine Beschreibung für den Dienst ein.
6. Gehen Sie im Abschnitt Einstellungen für das Service-Repository wie folgt vor:

   a. Wählen Sie unter CodeStar Verbindung Ihre Verbindung aus der Liste aus.

   b. Wählen Sie für Repository-ID den Namen Ihres Quellcode-Repositorys aus der Liste aus.

c. Wählen Sie unter Branch-Name den Namen Ihres Quellcode-Repository-Branches aus der Liste aus.

7. (Optional) Wählen Sie im Abschnitt „Tags“ die Option Neues Tag hinzufügen aus und geben Sie einen Schlüssel und einen Wert ein, um ein vom Kunden verwaltetes Tag zu erstellen.
8. Wählen Sie Weiter aus.
9. Auf der Seite Benutzerdefinierte Einstellungen konfigurieren im Abschnitt Serviceinstanzen im Abschnitt Neue Instanz. Sie müssen Werte für die `required` Parameter eingeben. Sie können Werte für die `optional` Parameter eingeben oder die Standardwerte verwenden, sofern diese angegeben sind.
10. Im Abschnitt Pipeline-Eingaben müssen Sie Werte für die `required` Parameter eingeben. Sie können Werte für die `optional` Parameter eingeben oder die Standardwerte verwenden, sofern diese angegeben sind.
11. Wählen Sie Weiter und überprüfen Sie Ihre Eingaben.
12. Wählen Sie Erstellen aus.

    Sehen Sie sich die Servicedetails und den Status sowie die AWS verwalteten und vom Kunden verwalteten Tags für Ihren Service an.
13. Wählen Sie im Navigationsbereich Services.

    Auf einer neuen Seite wird eine Liste Ihrer Dienste zusammen mit dem Status und anderen Servicedetails angezeigt.

AWS CLI

Wenn Sie den verwenden AWS CLI, geben Sie Diensteingaben in einer YAML-formatierten `spec` Datei an`.aws-proton/service.yaml`, die sich in Ihrem Quellcodeverzeichnis befindet.

Sie können den `get-service-template-minor-version` CLI-Befehl verwenden, um die für das Schema erforderlichen und optionalen Parameter anzuzeigen, für die Sie Werte in Ihrer Spezifikationsdatei angeben.

Wenn Sie eine Dienstvorlage verwenden möchten, die über Folgendes verfügt`pipelineProvisioning: "CUSTOMER_MANAGED"`, nehmen Sie den *`pipeline:`* Abschnitt nicht in Ihre Spezifikation auf und nehmen Sie keine `-repository-connection-arn-repository-id`, und `-branch-name` Parameter in Ihren `create-service` Befehl auf.

Erstellen Sie einen Dienst mit einer Service-Pipeline, wie in den folgenden CLI-Schritten gezeigt.

1. Richten Sie die Servicerolle für die Pipeline ein, wie im folgenden CLI-Beispielbefehl gezeigt.

   Befehl:

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

2. Die folgende Liste zeigt eine Beispielspezifikation, die auf dem Dienstvorlagenschema basiert und die Eingaben für die Service-Pipeline und die Instanzeingaben enthält.

   Spezifikation:

   ```
   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"
   ```

   Erstellen Sie einen Dienst mit einer Pipeline, wie im folgenden CLI-Beispielbefehl und einer Antwort gezeigt.

   Befehl:

   ```
   $ 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"
   ```

   Antwort:

```
{
    "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"
    }
}
```

Erstellen Sie einen Dienst ohne Service-Pipeline, wie im folgenden CLI-Beispielbefehl und -antwort gezeigt.

Im Folgenden wird eine Beispielspezifikation gezeigt, die keine Eingaben für die Service-Pipeline enthält.

Spezifikation:

```
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"
```

Um einen Dienst ohne bereitgestellte Dienstpipeline zu erstellen, geben Sie den Pfad zu einer an **`spec.yaml`** und fügen keine Repository-Parameter hinzu, wie im folgenden CLI-Beispielbefehl und der folgenden Befehlsantwort gezeigt.

Befehl:

```
$ aws proton create-service \
        --name "MySimpleServiceNoPipeline" \
        --template-major-version "1" \
```

```
        --template-name "fargate-service" \
        --spec "file://spec-no-pipeline.yaml"
```

Antwort:

```
{
    "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"
    }
}
```

Servicedaten anzeigen

Sie können Servicedetails über die AWS Proton Konsole oder die anzeigen und auflisten AWS CLI.

AWS-Managementkonsole

Dienstdetails mithilfe der AWS Proton Konsole auflisten und anzeigen, wie in den folgenden Schritten gezeigt.

1. Um eine Liste Ihrer Dienste anzuzeigen, wählen Sie im Navigationsbereich Dienste aus.
2. Um Detaildaten anzuzeigen, wählen Sie den Namen eines Dienstes aus.

   Zeigen Sie Ihre Servicedetails an.

AWS CLI

Zeigen Sie die Details eines Dienstes mit einer Service-Pipeline an, wie im folgenden CLI-Beispielbefehl und -antwort gezeigt.

Befehl:

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

```
    --name "simple-svc"
```

Antwort:

```
{
    "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"
    }
}
```

Zeigen Sie die Details eines Dienstes ohne Service-Pipeline an, wie im folgenden CLI-Beispielbefehl und der folgenden Antwort dargestellt.

Befehl:

```
$ aws proton get-service \
    --name "simple-svc-no-pipeline"
```

Antwort:

```
{
    "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"
    }
}
```

Bearbeiten Sie einen Dienst

Sie können die folgenden Änderungen an einem AWS Proton Dienst vornehmen.

- Bearbeiten Sie die Dienstbeschreibung.
- Bearbeiten Sie einen Dienst, indem Sie Dienstinstanzen hinzufügen und entfernen.

Dienstbeschreibung bearbeiten

Sie können die Konsole oder die verwenden AWS CLI , um eine Servicebeschreibung zu bearbeiten.

AWS-Managementkonsole

Bearbeiten Sie einen Dienst mithilfe der Konsole, wie in den folgenden Schritten beschrieben.

In der Liste der Dienste.

1. Wählen Sie in der AWS Proton Konsole Dienste aus.

2. Wählen Sie in der Liste der Dienste das Optionsfeld links neben dem Dienst aus, den Sie aktualisieren möchten.
3. Wählen Sie Bearbeiten aus.
4. Füllen Sie auf der Seite Dienst konfigurieren das Formular aus und wählen Sie Weiter.
5. Wählen Sie auf der Seite Benutzerdefinierte Einstellungen konfigurieren die Option Weiter aus.
6. Überprüfen Sie Ihre Änderungen und wählen Sie Änderungen speichern.

Auf der Seite mit den Servicedetails.

1. Wählen Sie in der AWS Proton Konsole Dienste aus.
2. Wählen Sie in der Liste der Dienste den Namen des Dienstes aus, den Sie bearbeiten möchten.
3. Wählen Sie auf der Seite mit den Service-Details die Option Bearbeiten aus.
4. Füllen Sie auf der Seite Service konfigurieren das Formular aus und wählen Sie Weiter.
5. Füllen Sie auf der Seite Benutzerdefinierte Einstellungen konfigurieren das Formular aus und wählen Sie Weiter.
6. Überprüfen Sie Ihre Änderungen und wählen Sie Änderungen speichern.

AWS CLI

Bearbeiten Sie eine Beschreibung wie im folgenden CLI-Beispielbefehl und der folgenden Antwort gezeigt.

Befehl:

```
$ aws proton update-service \
    --name "MySimpleService" \
    --description "Edit by updating description"
```

Antwort:

```
{
    "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"
    }
}
```

Bearbeiten Sie einen Dienst, um Dienstinstanzen hinzuzufügen oder zu entfernen

Für einen AWS Proton Service können Sie Dienstinstanzen hinzufügen oder löschen, indem Sie eine bearbeitete Spezifikation einreichen. Die folgenden Bedingungen müssen für eine erfolgreiche Anfrage erfüllt sein:

- Ihr Service und Ihre Pipeline werden nicht bereits bearbeitet oder gelöscht, wenn Sie die Bearbeitungsanfrage einreichen.
- Ihre bearbeitete Spezifikation enthält keine Änderungen, die die Service-Pipeline ändern, oder Änderungen an vorhandenen Serviceinstanzen, die nicht gelöscht werden sollen.
- Ihre bearbeitete Spezifikation entfernt keine vorhandene Dienstinstanz, der eine Komponente angehängt ist. Um eine solche Dienstinstanz zu löschen, sollten Sie zuerst die Komponente aktualisieren, um sie von ihrer Dienstinstanz zu trennen. Weitere Hinweise zu Komponenten finden Sie unterKomponenten.

Instanzen, bei denen beim Löschen ein Fehler aufgetreten ist, sind Dienstinstanzen im `DELETE_FAILED` Status. Wenn Sie eine Servicebearbeitung anfordern, wird AWS Proton versucht, die beim Löschen fehlgeschlagenen Instanzen im Rahmen des Bearbeitungsvorgangs für Sie zu entfernen. Wenn eine Ihrer Dienstinstanzen nicht gelöscht werden konnte, sind den Instanzen möglicherweise immer noch Ressourcen zugeordnet, auch wenn sie in der Konsole oder nicht sichtbar sind. AWS CLIÜberprüfen Sie die Infrastrukturressourcen Ihrer Instances, bei denen das Löschen fehlgeschlagen ist, und bereinigen Sie sie, damit wir sie für Sie entfernen AWS Proton können.

Informationen zum Kontingent an Dienstinstanzen für einen Service finden Sie unter. AWS Proton Kontingente Sie müssen außerdem mindestens eine Dienstinstanz für Ihren Service verwalten, nachdem er erstellt wurde. Zählt während des Aktualisierungsvorgangs die vorhandenen Dienstinstanzen und die Instanzen, die hinzugefügt oder entfernt werden sollen. AWS Proton Instanzen, bei denen das Löschen fehlschlug, sind in dieser Anzahl enthalten, und Sie müssen sie berücksichtigen, wenn Sie Ihre bearbeiten. `spec`

Verwenden Sie die Konsole oder AWS CLI um Serviceinstanzen hinzuzufügen oder zu entfernen

AWS-Managementkonsole

Bearbeiten Sie Ihren Service, um Dienstinstanzen mithilfe der Konsole hinzuzufügen oder zu entfernen.

In der AWS Proton Konsole

1. Wählen Sie im Navigationsbereich Services.
2. Wählen Sie den Dienst aus, den Sie bearbeiten möchten.
3. Wählen Sie Bearbeiten aus.
4. (Optional) Bearbeiten Sie auf der Seite Dienst konfigurieren den Dienstnamen oder die Beschreibung und wählen Sie dann Weiter aus.
5. Wählen Sie auf der Seite Benutzerdefinierte Einstellungen konfigurieren die Option Löschen aus, um eine Dienstinstanz zu löschen, und wählen Sie Neue Instanz hinzufügen, um eine Dienstinstanz hinzuzufügen und das Formular auszufüllen.
6. Wählen Sie Weiter aus.
7. Überprüfen Sie Ihr Update und wählen Sie Änderungen speichern.
8. In einem Modal werden Sie aufgefordert, das Löschen von Dienstinstanzen zu überprüfen. Folgen Sie den Anweisungen und wählen Sie Ja, löschen.
9. Sehen Sie sich auf der Seite mit den Servicedetails die Statusdetails für Ihren Dienst an.

AWS CLI

Fügen Sie Dienstinstanzen mit bearbeiteten Befehlen und Antworten hinzu und löschen Sie sie, **`spec`** wie in den folgenden AWS CLI Beispielbefehlen und Antworten gezeigt.

Wenn Sie die CLI verwenden, `spec` müssen Sie die zu löschenden Dienstinstanzen ausschließen und sowohl die hinzuzufügenden Dienstinstanzen als auch die vorhandenen Serviceinstanzen einbeziehen, die Sie nicht zum Löschen markiert haben.

Die folgende Liste zeigt das Beispiel `spec` vor der Bearbeitung und eine Liste der Serviceinstanzen, die von der Spezifikation bereitgestellt wurden. Diese Spezifikation wurde im vorherigen Beispiel für die Bearbeitung einer Servicebeschreibung verwendet.

Spezifikation:

```
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"
```

Der folgende `list-service-instances` CLI-Beispielbefehl und die folgende Antwort zeigen die aktiven Instanzen vor dem Hinzufügen oder Löschen einer Dienstinstanz.

Befehl:

```
$ aws proton list-service-instances \
    --service-name "MySimpleService"
```

Antwort:

```
{
    "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"
        }
    ]
}
```

Die folgende Liste zeigt das bearbeitete Beispiel, das zum Löschen und Hinzufügen einer Instanz `spec` verwendet wurde. Die bestehende Instanz mit dem Namen `my-instance` wird entfernt und eine neue Instanz mit dem Namen `yet-another-instance` hinzugefügt.

Spezifikation:

```
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"
```

Sie können `"${Proton::CURRENT_VAL}"` damit angeben, welche Parameterwerte aus dem Original beibehalten werden sollen`spec`, sofern die Werte in der `spec` vorhanden sind. `get-service`Dient zum Anzeigen des Originals `spec` für einen Service, wie unter beschriebenServicedaten anzeigen.

Die folgende Liste zeigt, wie Sie sicherstellen können, dass Sie `spec` keine Änderungen `"${Proton::CURRENT_VAL}"` an den Parameterwerten enthalten, damit die vorhandenen Services-Instanzen erhalten bleiben.

Spezifikation:

```
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"
```

Die nächste Liste zeigt den CLI-Befehl und die Antwort zum Bearbeiten des Dienstes.

Befehl:

```
$ aws proton update-service
```

```
    --name "MySimpleService" \
    --description "Edit by adding and deleting a service instance" \
    --spec "file://spec.yaml"
```

Antwort:

```
{
    "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"
    }
}
```

Der folgende `list-service-instances` Befehl und die folgende Antwort bestätigen, dass die vorhandene Instanz mit dem Namen `my-instance` entfernt und eine neue Instanz mit dem Namen `yet-another-instance` hinzugefügt wurde.

Befehl:

```
$ aws proton list-service-instances \
    --service-name "MySimpleService"
```

Antwort:

```
{
    "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"
        }
    ]
}
```

Was passiert, wenn Sie Dienstinstanzen hinzufügen oder entfernen

Nachdem Sie eine Servicebearbeitung zum Löschen und Hinzufügen von Dienstinstanzen eingereicht haben, AWS Proton werden die folgenden Aktionen ausgeführt.

- Legt den Dienst auf fest`UPDATE_IN_PROGRESS`.
- Wenn der Dienst über eine Pipeline verfügt, legt dessen Status auf Pipeline-Aktionen fest `IN_PROGRESS` und blockiert sie.
- Legt alle Dienstinstanzen fest, die gelöscht werden sollen`DELETE_IN_PROGRESS`.
- Blockiert Dienstaktionen.
- Blockiert Aktionen auf Dienstinstanzen, die zum Löschen markiert sind.
- Erzeugt neue Dienstinstanzen.
- Löscht Instanzen, die Sie zum Löschen aufgelistet haben.
- Versuche, beim Löschen fehlgeschlagene Instanzen zu entfernen.

- Wenn die Hinzufügungen und Löschungen abgeschlossen sind, wird die Service-Pipeline (falls vorhanden) erneut bereitgestellt, Ihr Service auf Service- und Pipeline-Aktionen `ACTIVE` gesetzt und aktiviert.

AWS Proton versucht, Fehlermodi wie folgt zu beheben.

- Wenn eine oder mehrere Dienstinstanzen nicht erstellt werden konnten, wird AWS Proton versucht, die Bereitstellung aller neu erstellten Dienstinstanzen aufzuheben, und der `spec` vorherige Zustand wird wiederhergestellt. Es löscht keine Dienstinstanzen und ändert die Pipeline in keiner Weise.
- Wenn eine oder mehrere Dienstinstanzen nicht gelöscht werden konnten, wird die Pipeline ohne die gelöschten Instanzen AWS Proton erneut bereitgestellt. Die `spec` wird aktualisiert, um die hinzugefügten Instanzen einzubeziehen und die Instanzen auszuschließen, die zum Löschen markiert wurden.
- Wenn die Bereitstellung der Pipeline fehlschlägt, wird kein Rollback versucht, und sowohl der Dienst als auch die Pipeline geben den Status eines fehlgeschlagenen Updates wieder.

Tagging und Änderungen am Dienst

Wenn Sie im Rahmen Ihrer Servicebearbeitung Dienstinstanzen hinzufügen, werden AWS verwaltete Tags auf die neuen Instanzen und bereitgestellten Ressourcen übertragen und automatisch für diese erstellt. Wenn Sie neue Tags erstellen, werden diese Tags nur auf die neuen Instanzen angewendet. Bestehende, vom Kunden verwaltete Service-Tags werden ebenfalls auf die neuen Instanzen übertragen. Weitere Informationen finden Sie unter AWS Proton Ressourcen und Tagging.

Einen Service löschen

Sie können einen AWS Proton Dienst mit seinen Instanzen und seiner Pipeline mithilfe der AWS Proton Konsole oder der AWS CLI löschen.

Sie können keinen Dienst löschen, der über eine Dienstinstanz mit einer angehängten Komponente verfügt. Um einen solchen Dienst zu löschen, sollten Sie zunächst alle angehängten Komponenten aktualisieren, um sie von ihren Dienstinstanzen zu trennen. Weitere Informationen zu Komponenten finden Sie unterKomponenten.

AWS-Managementkonsole

Löschen Sie einen Dienst mithilfe der Konsole, wie in den folgenden Schritten beschrieben.

Auf der Seite mit den Servicedetails.

1. Wählen Sie in der AWS Proton Konsole Dienste aus.
2. Wählen Sie in der Liste der Dienste den Namen des Dienstes aus, den Sie löschen möchten.
3. Wählen Sie auf der Service-Detailseite Aktionen und anschließend Löschen aus.
4. In einem Modalfenster werden Sie aufgefordert, die Löschaktion zu bestätigen.
5. Folgen Sie den Anweisungen und wählen Sie Ja, löschen.

AWS CLI

Löschen Sie einen Dienst, wie im folgenden CLI-Beispielbefehl und der folgenden Antwort gezeigt.

Befehl:

```
$ aws proton delete-service \
    --name "simple-svc"
```

Antwort:

```
{
    "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"
    }
}
```

Dienstinstanzdaten anzeigen

Erfahren Sie, wie Sie Detaildaten zu AWS Proton Serviceinstanzen anzeigen. Sie können die Konsole oder die verwenden AWS CLI.

Eine Dienstinstanz gehört zu einem Dienst. Sie können eine Instanz nur im Rahmen von Aktionen zum Bearbeiten, Erstellen und Löschen von Diensten erstellen oder löschen. Informationen zum Hinzufügen und Entfernen von Instanzen zu einem Service finden Sie unterBearbeiten Sie einen Dienst.

AWS-Managementkonsole

Mithilfe der AWS Proton Konsole können Sie Details zu Dienstinstanzen auflisten und anzeigen, wie in den folgenden Schritten gezeigt.

1. Um eine Liste Ihrer Dienstinstanzen anzuzeigen, wählen Sie im Navigationsbereich Services-Instanzen aus.
2. Um Detaildaten anzuzeigen, wählen Sie den Namen einer Dienstinstanz.

   Zeigen Sie die Detaildaten Ihrer Serviceinstanz an.

AWS CLI

Listet die Details der Dienstinstanz auf und zeigt sie an, wie in den folgenden CLI-Beispielbefehlen und -antworten gezeigt.

Befehl:

```
$ aws proton list-service-instances
```

Antwort:

```
{
    "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"
        }
    ]
}
```

Befehl:

```
$ aws proton get-service-instance \
    --name "instance-one" \
    --service-name "simple-svc"
```

Antwort:

```
{
    "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"
    }
}
```

Aktualisieren Sie eine Dienstinstanz

Erfahren Sie, wie Sie eine AWS Proton Dienstinstanz aktualisieren und das Update abbrechen.

Eine Dienstinstanz gehört zu einem Dienst. Sie können eine Instanz nur im Rahmen von Aktionen zum Bearbeiten, Erstellen und Löschen von Diensten erstellen oder löschen. Informationen zum Hinzufügen und Entfernen von Instanzen zu einem Service finden Sie unterBearbeiten Sie einen Dienst.

Es gibt vier Modi zum Aktualisieren einer Dienstinstanz, wie in der folgenden Liste beschrieben. Bei Verwendung von definiert das `deployment-type` Feld den Modus. AWS CLI Wenn Sie die Konsole verwenden, sind diese Modi den Aktionen Bearbeiten und Auf neueste Nebenversion aktualisieren und Auf neueste Hauptversion aktualisieren zugeordnet, die auf der Detailseite der Service-Instanz im Dropdownmenü Aktionen angezeigt werden.

`NONE`

In diesem Modus findet keine Bereitstellung statt. Nur die angeforderten Metadatenparameter werden aktualisiert.

`CURRENT_VERSION`

In diesem Modus wird die Dienstinstanz bereitgestellt und mit der neuen Spezifikation, die Sie angeben, aktualisiert. Nur die angeforderten Parameter werden aktualisiert. Geben Sie keine Parameter für Neben- oder Hauptversionen an, wenn Sie dies verwenden`deployment-type`.

`MINOR_VERSION`

In diesem Modus wird die Dienstinstanz bereitgestellt und standardmäßig mit der veröffentlichten, empfohlenen (neuesten) Nebenversion der aktuell verwendeten Hauptversion aktualisiert. Sie können auch eine andere Nebenversion der aktuell verwendeten Hauptversion angeben.

`MAJOR_VERSION`

In diesem Modus wird die Dienstinstanz standardmäßig bereitgestellt und mit der veröffentlichten, empfohlenen (neuesten) Haupt- und Nebenversion der aktuellen Vorlage aktualisiert. Sie können auch eine andere Hauptversion angeben, die höher ist als die verwendete Hauptversion und eine Nebenversion (optional).

Sie können versuchen, die Bereitstellung eines Service Instance-Updates abzubrechen, falls dies der Fall `deploymentStatus` ist`IN_PROGRESS`. AWS Proton versucht, die Bereitstellung abzubrechen. Eine erfolgreiche Stornierung ist nicht garantiert.

Wenn Sie eine Update-Bereitstellung stornieren, wird AWS Proton versucht, die Bereitstellung wie in den folgenden Schritten beschrieben abzubrechen.

- Setzt den Bereitstellungsstatus auf`CANCELLING`.
- Stoppt die laufende Bereitstellung und löscht alle neuen Ressourcen, die durch die Bereitstellung erstellt wurden, wann`IN_PROGRESS`.
- Setzt den Bereitstellungsstatus auf`CANCELLED`.
- Setzt den Zustand der Ressource auf den Zustand vor dem Start der Bereitstellung zurück.

Weitere Informationen zum Abbrechen einer Service-Instance-Bereitstellung finden Sie CancelServiceInstanceDeploymentin der AWS Proton API-Referenz.

Verwenden Sie die Konsole oder AWS CLI um Aktualisierungen vorzunehmen oder Update-Bereitstellungen abzubrechen.

AWS-Managementkonsole

Aktualisieren Sie eine Dienstinstanz mithilfe der Konsole, indem Sie die folgenden Schritte ausführen.

1. Wählen Sie in der AWS Proton Konsole im Navigationsbereich Service Instances aus.
2. Wählen Sie in der Liste der Dienstinstanzen den Namen der Dienstinstanz aus, die Sie aktualisieren möchten.
3. Wählen Sie Aktionen und dann eine der Aktualisierungsoptionen, Bearbeiten, um die Spezifikation oder Aktionen zu aktualisieren, und dann Auf die neueste Nebenversion aktualisieren oder Auf die neueste Hauptversion aktualisieren.
4. Füllen Sie jedes Formular aus und klicken Sie auf Weiter, bis Sie zur Seite „Überprüfen“ gelangen.
5. Überprüfe deine Änderungen und wähle „Aktualisieren“.

AWS CLI

Aktualisieren Sie eine Dienstinstanz auf eine neue Nebenversion, wie in den CLI-Beispielbefehlen und -antworten gezeigt.

Wenn Sie Ihre Serviceinstanz mit einer geänderten Version aktualisieren`spec`, können Sie `"${Proton::CURRENT_VAL}"` damit angeben, welche Parameterwerte gegenüber dem Original beibehalten werden sollen`spec`, sofern die Werte in der vorhanden sind`spec`. `get-service`Dient zum Anzeigen des Originals `spec` für eine Serviceinstanz, wie unter beschriebenServicedaten anzeigen.

Das folgende Beispiel zeigt, wie Sie `"${Proton::CURRENT_VAL}"` in a verwenden können`spec`.

Spezifikation:

```
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"
```

Befehl: zum Aktualisieren

```
$ 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"
```

Antwort:

```
{
    "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"
    }
}
```

Befehl: um den Status abzurufen und zu bestätigen

```
$ aws proton get-service-instance \
    --name "instance-one" \
    --service-name "simple-svc"
```

Antwort:

```
{
    "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"
    }
}
```

AWS-Managementkonsole

Brechen Sie die Bereitstellung einer Service-Instanz mithilfe der Konsole ab, wie in den folgenden Schritten gezeigt.

1. Wählen Sie in der AWS Proton Konsole im Navigationsbereich Service Instances aus.
2. Wählen Sie in der Liste der Dienstinstanzen den Namen der Dienstinstanz mit dem Bereitstellungsupdate aus, das Sie stornieren möchten.
3. Wenn der Status der Update-Bereitstellung auf In Bearbeitung steht, wählen Sie auf der Detailseite der Service-Instanz Aktionen und dann Bereitstellung abbrechen aus.
4. In einem Modalfenster werden Sie aufgefordert, die Stornierung zu bestätigen. Wählen Sie Bereitstellung abbrechen aus.
5. Ihr Status für die Bereitstellung von Updates ist auf Storniert und dann auf Storniert gesetzt, um die Stornierung abzuschließen.

AWS CLI

Brechen Sie ein Update zur Bereitstellung einer IN_PROGRESS-Dienstinstanz auf die neue Nebenversion 2 ab, wie in den folgenden CLI-Beispielbefehlen und -antworten gezeigt.

In der für dieses Beispiel verwendeten Vorlage ist eine Wartebedingung enthalten, sodass der Abbruch beginnt, bevor die Bereitstellung des Updates erfolgreich ist.

Befehl: zum Abbrechen

```
$ aws proton cancel-service-instance-deployment \
    --service-instance-name "instance-one" \
    --service-name "simple-svc"
```

Antwort:


```
{
    "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"
    }
}
```


Befehl: um den Status abzurufen und zu bestätigen

```
$ aws proton get-service-instance \
    --name "instance-one" \
    --service-name "simple-svc"
```

Antwort:


```
{
    "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"
    }
}
```


Aktualisieren Sie eine Service-Pipeline

Erfahren Sie, wie Sie eine AWS Proton Service-Pipeline aktualisieren und das Update abbrechen.

Eine Service-Pipeline gehört zu einem Service. Sie können eine Pipeline nur im Rahmen von Aktionen zum Erstellen und Löschen von Diensten erstellen oder löschen.

Es gibt vier Modi zum Aktualisieren einer Service-Pipeline, wie in der folgenden Liste beschrieben. Bei Verwendung von definiert das `deployment-type` Feld den Modus. AWS CLI Wenn Sie die Konsole verwenden, werden diese Modi der Pipeline „Bearbeiten“ und „Auf empfohlene Version aktualisieren“ zugeordnet.

`NONE`

In diesem Modus findet keine Bereitstellung statt. Nur die angeforderten Metadatenparameter werden aktualisiert.

`CURRENT_VERSION`

In diesem Modus wird die Service-Pipeline bereitgestellt und mit der neuen Spezifikation, die Sie bereitstellen, aktualisiert. Nur die angeforderten Parameter werden aktualisiert. Geben Sie keine Parameter für Neben- oder Hauptversionen an, wenn Sie dies verwenden`deployment-type`.

`MINOR_VERSION`

In diesem Modus wird die Service-Pipeline bereitgestellt und standardmäßig mit der veröffentlichten, empfohlenen (neuesten) Nebenversion der aktuell verwendeten Hauptversion aktualisiert. Sie können auch eine andere Nebenversion der aktuell verwendeten Hauptversion angeben.

`MAJOR_VERSION`

In diesem Modus wird die Service-Pipeline standardmäßig bereitgestellt und mit der veröffentlichten, empfohlenen (neuesten) Haupt- und Nebenversion der aktuellen Vorlage aktualisiert. Sie können auch eine andere Hauptversion angeben, die höher ist als die verwendete Hauptversion und eine Nebenversion (optional).

Sie können versuchen, die Bereitstellung eines Service Pipeline-Updates abzubrechen, falls dies der `deploymentStatus` Fall ist`IN_PROGRESS`. AWS Proton versucht, die Bereitstellung abzubrechen. Eine erfolgreiche Stornierung ist nicht garantiert.

Wenn Sie eine Update-Bereitstellung stornieren, wird AWS Proton versucht, die Bereitstellung wie in den folgenden Schritten beschrieben abzubrechen.

- Setzt den Bereitstellungsstatus auf`CANCELLING`.
- Stoppt die laufende Bereitstellung und löscht alle neuen Ressourcen, die durch die Bereitstellung erstellt wurden, wann`IN_PROGRESS`.
- Setzt den Bereitstellungsstatus auf`CANCELLED`.
- Setzt den Zustand der Ressource auf den Zustand vor dem Start der Bereitstellung zurück.

Weitere Informationen zum Stornieren einer Service-Pipeline-Bereitstellung finden Sie CancelServicePipelineDeploymentin der AWS Proton API-Referenz.

Verwenden Sie die Konsole oder AWS CLI um Aktualisierungen vorzunehmen oder Update-Bereitstellungen abzubrechen.

AWS-Managementkonsole

Aktualisieren Sie eine Service-Pipeline mithilfe der Konsole, wie in den folgenden Schritten beschrieben.

1. Wählen Sie in der AWS Proton Konsole Dienste aus.
2. Wählen Sie in der Liste der Dienste den Namen des Dienstes aus, für den Sie die Pipeline aktualisieren möchten.
3. Auf der Service-Detailseite gibt es zwei Registerkarten: Übersicht und Pipeline. Wählen Sie Pipeline.
4. Wenn Sie die Spezifikationen aktualisieren möchten, wählen Sie Pipeline bearbeiten und füllen Sie jedes Formular aus. Wählen Sie Weiter, bis Sie das endgültige Formular ausgefüllt haben, und wählen Sie dann Pipeline aktualisieren.

   Wenn Sie auf eine neue Version aktualisieren möchten und unter Pipeline-Vorlage ein Informationssymbol angezeigt wird, das darauf hinweist, dass eine neue Version verfügbar ist, wählen Sie den Namen der neuen Vorlagenversion aus.

   a. Wählen Sie Auf empfohlene Version aktualisieren.
   b. Füllen Sie jedes Formular aus und klicken Sie auf Weiter, bis Sie das endgültige Formular ausgefüllt haben, und wählen Sie dann Aktualisieren.

AWS CLI

Aktualisieren Sie eine Service-Pipeline auf eine neue Nebenversion, wie in den folgenden CLI-Beispielbefehlen und -antworten gezeigt.

Wenn Sie Ihre Service-Pipeline mit einer geänderten Version aktualisieren`spec`, können Sie `"${Proton::CURRENT_VAL}"` damit angeben, welche Parameterwerte gegenüber dem Original beibehalten werden sollen`spec`, sofern die Werte in der vorhanden sind`spec`. `get-service`Dient zum Anzeigen des Originals `spec` für eine Service-Pipeline, wie unter beschriebenServicedaten anzeigen.

Das folgende Beispiel zeigt, wie Sie `"${Proton::CURRENT_VAL}"` in a verwenden können`spec`.

Spezifikation:

```
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"
```

Befehl: zum Aktualisieren

```
$ 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"
```

Antwort:

```
{
    "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"
    }
}
```

Befehl: um den Status abzurufen und zu bestätigen

```
$ aws proton get-service \
    --name "simple-svc"
```

Antwort:


```
{
    "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"
    }
}
```

AWS-Managementkonsole

Brechen Sie eine Service-Pipeline-Bereitstellung mithilfe der Konsole ab, wie in den folgenden Schritten gezeigt.

1. Wählen Sie in der AWS Proton Konsole im Navigationsbereich Dienste aus.
2. Wählen Sie in der Liste der Dienste den Namen des Dienstes aus, der über die Pipeline mit dem Bereitstellungsupdate verfügt, das Sie stornieren möchten.
3. Wählen Sie auf der Service-Detailseite die Registerkarte Pipeline aus.
4. Wenn der Status der Aktualisierungsbereitstellung „In Bearbeitung“ lautet, wählen Sie auf der Seite mit den Servicepipeline-Details die Option Bereitstellung abbrechen aus.
5. In einem Fenster werden Sie aufgefordert, die Stornierung zu bestätigen. Wählen Sie Bereitstellung abbrechen aus.
6. Ihr Status für die Bereitstellung von Updates ist auf Storniert und dann auf Storniert gesetzt, um die Stornierung abzuschließen.

AWS CLI

Brechen Sie ein IN_PROGRESS-Dienst-Pipeline-Bereitstellungsupdate auf Nebenversion 2 ab, wie in den folgenden CLI-Beispielbefehlen und -antworten gezeigt.

In der für dieses Beispiel verwendeten Vorlage ist eine Wartebedingung enthalten, sodass der Abbruch beginnt, bevor die Bereitstellung des Updates erfolgreich ist.

Befehl: zum Abbrechen

```
$ aws proton cancel-service-pipeline-deployment \
    --service-name "simple-svc"
```

Antwort:

```
{
    "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"
    }
}
```

Befehl: um den Status abzurufen und zu bestätigen

```
$ aws proton get-service \
    --name "simple-svc"
```

Antwort:

```
{
    "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 Komponenten

Komponenten sind eine Art von Ressource. AWS Proton Sie verleihen Servicevorlagen mehr Flexibilität. Komponenten bieten Plattformteams einen Mechanismus zur Erweiterung der wichtigsten Infrastrukturmuster und zur Definition von Schutzmaßnahmen, die es Entwicklern ermöglichen, Aspekte ihrer Anwendungsinfrastruktur zu verwalten.

Darin definieren AWS Proton Administratoren eine Standardinfrastruktur, die von Entwicklungsteams und Anwendungen verwendet wird. Entwicklungsteams müssen jedoch möglicherweise zusätzliche Ressourcen für ihre spezifischen Anwendungsfälle hinzufügen, z. B. Amazon Simple Queue Service (Amazon SQS) -Warteschlangen oder Amazon DynamoDB-Tabellen. Diese anwendungsspezifischen Ressourcen können sich häufig ändern, insbesondere während der frühen Anwendungsentwicklung. Die Beibehaltung dieser häufigen Änderungen in vom Administrator erstellten Vorlagen könnte schwierig zu verwalten und zu skalieren sein — Administratoren müssten viel mehr Vorlagen verwalten, ohne dass der Administrator einen echten Mehrwert hätte. Die Alternative, Anwendungsentwickler Vorlagen für ihre Anwendungen erstellen zu lassen, ist ebenfalls nicht ideal, da sie Administratoren die Möglichkeit nimmt, die wichtigsten Architekturkomponenten wie Aufgaben zu standardisieren. AWS Fargate An dieser Stelle kommen Komponenten ins Spiel.

Mit einer Komponente kann ein Entwickler seiner Anwendung zusätzliche Ressourcen hinzufügen, die über das hinausgehen, was Administratoren in Umgebungs- und Dienstvorlagen definiert haben. Der Entwickler fügt die Komponente dann einer Dienstinstanz hinzu. AWS Proton stellt durch die Komponente definierte Infrastrukturressourcen genauso bereit, wie sie Ressourcen für Umgebungen und Dienstinstanzen bereitstellt.

Eine Komponente kann Eingaben von Serviceinstanzen lesen und Ausgaben für die Dienstinstanz bereitstellen, was für ein vollständig integriertes Erlebnis sorgt. Wenn die Komponente beispielsweise einen Amazon Simple Storage Service (Amazon S3) -Bucket zur Verwendung durch eine Service-Instance hinzufügt, kann die Komponentenvorlage die Umgebungs- und Service-Instance-Namen bei der Benennung des Buckets berücksichtigen. Wenn die Service-Vorlage zur Bereitstellung einer Service-Instance AWS Proton gerendert wird, kann die Service-Instance auf den Bucket verweisen und ihn verwenden.

Bei den Komponenten, die AWS Proton derzeit unterstützt werden, handelt es sich um direkt definierte Komponenten. Sie übergeben die Infrastructure as Code (IaC) -Datei, die die Infrastruktur der Komponente definiert, direkt an die AWS Proton API oder Konsole. Dies unterscheidet sich von einer Umgebung oder einem Dienst, in dem Sie IaC in einem Vorlagenpaket definieren und das Paket

als Vorlagenressource registrieren und dann eine Vorlagenressource verwenden, um die Umgebung oder den Dienst zu erstellen.

> **Note**
>
> Direkt definierte Komponenten ermöglichen es Entwicklern, zusätzliche Infrastruktur zu definieren und bereitzustellen. AWS Proton stellt alle direkt definierten Komponenten bereit, die in derselben Umgebung ausgeführt werden und dieselbe AWS Identity and Access Management (IAM-) Rolle verwenden.

Ein Administrator kann auf zwei Arten steuern, was Entwickler mit Komponenten tun können:

- Unterstützte Komponentenquellen — Ein Administrator kann das Anhängen von Komponenten an Dienstinstanzen auf der Grundlage einer Eigenschaft von AWS Proton Dienstvorlagenversionen zulassen. Standardmäßig können Entwickler keine Komponenten an Serviceinstanzen anhängen.

  Weitere Informationen zu dieser Eigenschaft finden Sie unter dem supportedComponentSourcesParameter der CreateServiceTemplateVersionAPI-Aktion in der AWS Proton API-Referenz.

  > **Note**
  >
  > Wenn Sie die Vorlagensynchronisierung verwenden, werden Dienstvorlagenversionen implizit AWS Proton erstellt, wenn Sie Änderungen an einem Dienstvorlagenpaket in einem Repository festschreiben. In diesem Fall geben Sie diese Eigenschaft in einer Datei an, die jeder Hauptversion der Service-Vorlage zugeordnet ist, anstatt die unterstützten Komponentenquellen bei der Erstellung der Service-Vorlagenversion anzugeben. Weitere Informationen finden Sie unter the section called “Synchronisieren von Dienstvorlagen”.

- Komponentenrollen — Ein Administrator kann einer Umgebung eine Komponentenrolle zuweisen. AWS Proton übernimmt diese Rolle, wenn es eine Infrastruktur bereitstellt, die durch eine direkt definierte Komponente in der Umgebung definiert ist. Daher beschränkt sich die Komponentenrolle auf die Infrastruktur, die Entwickler mithilfe direkt definierter Komponenten in der Umgebung hinzufügen können. In Ermangelung der Komponentenrolle können Entwickler keine direkt definierten Komponenten in der Umgebung erstellen.

  Weitere Informationen zum Zuweisen einer Komponentenrolle finden Sie unter dem componentRoleArnParameter der CreateEnvironmentAPI-Aktion in der AWS Proton API-Referenz.

> **Note**
>
> Komponentenrollen werden in Selbstverwaltete Bereitstellung Umgebungen nicht verwendet.

Themen

Wie schneiden Komponenten im Vergleich zu anderen AWS Proton Ressourcen ab?

In vielerlei Hinsicht ähneln Komponenten anderen AWS Proton Ressourcen. Ihre Infrastruktur ist in einer IaC-Vorlagendatei definiert, die entweder im CloudFormation YAML- oder Terraform HCL-Format verfasst wurde. AWS Proton kann die Komponenteninfrastruktur entweder mithilfe einer verwalteten Bereitstellung oder einer selbstverwalteten Bereitstellung bereitstellen.AWS

Komponenten unterscheiden sich jedoch in einigen Punkten von anderen AWS Proton Ressourcen:

- Getrennter Zustand — Komponenten sind so konzipiert, dass sie an Serviceinstanzen angehängt werden und ihre Infrastruktur erweitern. Sie können sich aber auch in einem getrennten Zustand befinden, in dem sie an keine Dienstinstanz angehängt sind. Weitere Informationen zu Komponentenstatus finden Sie unterthe section called “Status der Komponenten”.
- Kein Schema — Komponenten haben kein zugeordnetes Schema, wie es bei Vorlagenpaketen der Fall ist. Komponenteneingaben werden durch einen Dienst definiert. Eine Komponente kann Eingaben verarbeiten, wenn sie an eine Dienstinstanz angehängt ist.
- Keine vom Kunden verwalteten Komponenten — AWS Proton stellt immer die Komponenteninfrastruktur für Sie bereit. Es gibt keine Version der Komponenten, bei denen Sie

Ihre eigenen Ressourcen mitbringen können. Weitere Informationen zu vom Kunden verwalteten Umgebungen finden Sie unterthe section called “Create”.

- Keine Vorlagenressource — Direkt definierten Komponenten ist keine Vorlagenressource zugeordnet, ähnlich wie bei Umgebungs- und Dienstvorlagen. Sie stellen der Komponente direkt eine IaC-Vorlagendatei zur Verfügung. In ähnlicher Weise stellen Sie direkt ein Manifest bereit, das die Vorlagensprache und die Rendering-Engine für die Bereitstellung der Infrastruktur der Komponente definiert. Sie erstellen die Vorlagendatei und das Manifest auf ähnliche Weise wie beim Erstellen eines Vorlagenpakets. Bei direkt definierten Komponenten ist es jedoch nicht erforderlich, IaC-Dateien als Bundles an bestimmten Speicherorten zu speichern, und Sie erstellen keine Vorlagenressource AWS Proton aus IaC-Dateien.
- Keine CodeBuild basierte Bereitstellung — Sie können direkt definierte Komponenten nicht mithilfe Ihres eigenen benutzerdefinierten Bereitstellungsskripts bereitstellen, das als basierte Bereitstellung bezeichnet wird. CodeBuild Weitere Informationen finden Sie unter the section called “CodeBuild Bereitstellung”.

Komponenten in der Konsole AWS Proton

Verwenden Sie die AWS Proton Konsole, um AWS Proton Komponenten zu erstellen, zu aktualisieren, anzuzeigen und zu verwenden.

Die folgenden Konsolenseiten beziehen sich auf Komponenten. Wir enthalten direkte Links zu Konsolenseiten der obersten Ebene.

- Komponenten — Sehen Sie sich die Liste der Komponenten in Ihrem AWS Konto an. Sie können neue Komponenten erstellen und vorhandene Komponenten aktualisieren oder löschen. Wählen Sie einen Komponentennamen in der Liste aus, um die zugehörige Detailseite aufzurufen.

  Ähnliche Listen gibt es auch auf den Seiten mit den Umgebungsdetails und den Serviceinstanzen. In diesen Listen werden nur die Komponenten angezeigt, die der aufgerufenen Ressource zugeordnet sind. Wenn Sie eine Komponente aus einer dieser Listen erstellen, AWS Proton wählt Sie die zugehörige Umgebung auf der Seite Komponente erstellen vorab aus.
- Komponentendetails — Um die Seite mit den Komponentendetails anzuzeigen, wählen Sie in der Komponentenliste einen Komponentennamen aus.

  Sehen Sie sich auf der Detailseite die Komponentendetails und den Status an und aktualisieren oder löschen Sie die Komponente. Sie können Listen mit Ausgaben (z. B. bereitgestellte

Ressourcen ARNs), bereitgestellten CloudFormation Stacks und zugewiesenen Tags anzeigen und verwalten.

- Komponente erstellen — Erstellen Sie eine Komponente. Geben Sie den Namen und die Beschreibung der Komponente ein, wählen Sie die zugehörigen Ressourcen aus, geben Sie die IaC-Quelldatei der Komponente an und weisen Sie Tags zu.
- Komponente aktualisieren — Um eine Komponente zu aktualisieren, wählen Sie die Komponente in der Komponentenliste aus, und klicken Sie dann im Menü Aktionen auf Komponente aktualisieren. Sie können auch auf den Seiten mit den Komponentendetails die Option Aktualisieren auswählen.

  Sie können die meisten Details der Komponente aktualisieren. Sie können den Komponentennamen nicht aktualisieren. Und Sie können wählen, ob Sie die Komponente nach einem erfolgreichen Update erneut bereitstellen möchten oder nicht.
- Umgebung konfigurieren — Wenn Sie eine Umgebung erstellen oder aktualisieren, können Sie eine Komponentenrolle angeben. Diese Rolle steuert die Fähigkeit, direkt definierte Komponenten in der Umgebung auszuführen, und bietet Berechtigungen für deren Bereitstellung.
- Neue Dienstvorlagenversion erstellen — Wenn Sie eine Dienstvorlagenversion erstellen, können Sie Unterstützte Komponentenquellen für die Vorlagenversion angeben. Dadurch wird die Fähigkeit gesteuert, Komponenten an Dienstinstanzen von Diensten anzuhängen, die auf dieser Vorlagenversion basieren.

Komponenten in der AWS Proton API und AWS CLI

Verwenden Sie die AWS Proton API oder, AWS CLI um AWS Proton Komponenten zu erstellen, zu aktualisieren, anzuzeigen und zu verwenden.

Mit den folgenden API-Aktionen werden AWS Proton Komponentenressourcen direkt verwaltet.

- CreateComponent— Erstellen Sie eine AWS Proton Komponente.
- DeleteComponent— Löscht eine AWS Proton Komponente.
- GetComponent— Ruft detaillierte Daten für eine Komponente ab.
- ListComponentOutputs— Ruft eine Liste der Ausgaben der Komponente Infrastructure as Code (IaC) ab.
- ListComponentProvisionedResources— Listet die bereitgestellten Ressourcen für eine Komponente mit Details auf.

- ListComponents— Listet Komponenten mit Übersichtsdaten auf. Sie können die Ergebnisliste nach Umgebung, Dienst oder einer einzelnen Dienstinstanz filtern.

Die folgenden API-Aktionen anderer AWS Proton Ressourcen haben einige Funktionen, die sich auf Komponenten beziehen.

- CreateEnvironment, UpdateEnvironment— Wird verwendet, `componentRoleArn` um den Amazon-Ressourcennamen (ARN) der IAM-Servicerolle anzugeben, die bei der Bereitstellung direkt definierter Komponenten in dieser Umgebung AWS Proton verwendet wird. Er bestimmt den Umfang der Infrastruktur, den eine direkt definierte Komponente bereitstellen kann.
- CreateServiceTemplateVersion— Wird verwendet`supportedComponentSources`, um unterstützte Komponentenquellen anzugeben. Komponenten mit unterstützten Quellen können auf der Grundlage dieser Dienstvorlagenversion an Dienstinstanzen angehängt werden.

Häufig gestellte Fragen zu Komponenten

Was ist der Lebenszyklus einer Komponente?

Komponenten können sich in einem angehängten oder abgetrennten Zustand befinden. Sie sind so konzipiert, dass sie an eine Dienstinstanz angehängt werden können und deren Infrastruktur in den meisten Fällen verbessern. Getrennte Komponenten befinden sich in einem Übergangszustand, der es Ihnen ermöglicht, eine Komponente kontrolliert und sicher zu löschen oder an eine andere Dienstinstanz anzuhängen. Weitere Informationen finden Sie unter the section called “Status der Komponenten”.

Warum kann ich meine angehängten Komponenten nicht löschen?

Lösung: Um eine angehängte Komponente zu löschen, aktualisieren Sie die Komponente, um sie von der Dienstinstanz zu trennen, überprüfen Sie die Stabilität der Dienstinstanz und löschen Sie dann die Komponente.

Warum ist das erforderlich? Angeschlossene Komponenten bieten zusätzliche Infrastruktur, die Ihre Anwendung zur Ausführung ihrer Laufzeitfunktionen benötigt. Die Dienstinstanz verwendet möglicherweise Komponentenausgaben, um Ressourcen dieser Infrastruktur zu erkennen und zu nutzen. Das Löschen der Komponente und damit das Entfernen ihrer Infrastrukturressourcen könnte die angehängte Dienstinstanz stören.

Als zusätzliche Sicherheitsmaßnahme AWS Proton müssen Sie die Komponente aktualisieren und sie von ihrer Dienstinstanz trennen, bevor Sie sie löschen können. Anschließend können Sie Ihre Dienstinstanz validieren, um sicherzustellen, dass sie weiterhin bereitgestellt wird und ordnungsgemäß funktioniert. Wenn Sie ein Problem feststellen, können Sie die Komponente schnell wieder an die Dienstinstanz anhängen und dann daran arbeiten, das Problem zu beheben. Wenn Sie sicher sind, dass Ihre Serviceinstanz frei von jeglicher Abhängigkeit von der Komponente ist, können Sie die Komponente problemlos löschen.

Warum kann ich die angehängte Serviceinstanz einer Komponente nicht direkt ändern?

Lösung: Um den Anhang zu ändern, aktualisieren Sie die Komponente, um sie von der Serviceinstanz zu trennen, überprüfen Sie die Stabilität der Komponente und der Dienstinstanz und hängen Sie die Komponente dann an die neue Dienstinstanz an.

Warum ist das erforderlich? Eine Komponente ist so konzipiert, dass sie an eine Dienstinstanz angehängt werden kann. Ihre Komponente verwendet möglicherweise Eingaben von Dienstinstanzen für die Benennung und Konfiguration von Infrastrukturressourcen. Das Ändern der angehängten Dienstinstanz könnte die Komponente stören (zusätzlich zu einer möglichen Unterbrechung der Dienstinstanz), wie in den vorherigen häufig gestellten Fragen, Warum kann ich meine angehängten Komponenten nicht löschen? beschrieben.). Dies kann beispielsweise dazu führen, dass Ressourcen, die in der IaC-Vorlage der Komponente definiert sind, umbenannt und möglicherweise sogar ersetzt werden.

Als zusätzliche Sicherheitsmaßnahme AWS Proton müssen Sie die Komponente aktualisieren und sie von ihrer Dienstinstanz trennen, bevor Sie sie an eine andere Dienstinstanz anhängen können. Anschließend können Sie die Stabilität sowohl der Komponente als auch der Dienstinstanz überprüfen, bevor Sie die Komponente an die neue Dienstinstanz anhängen.

Status der Komponenten

AWS Proton Komponenten können sich in zwei grundlegend unterschiedlichen Zuständen befinden:

- Angefügt — Die Komponente ist an eine Dienstinstanz angehängt. Sie definiert eine Infrastruktur, die die Laufzeitfunktionalität der Dienstinstanz unterstützt. Die Komponente erweitert die in Umgebungs- und Dienstvorlagen definierte Infrastruktur um eine vom Entwickler definierte Infrastruktur.

  Eine typische Komponente befindet sich während des größten Teils ihres Lebenszyklus im angehängten Zustand.

- Getrennt — Die Komponente ist einer AWS Proton Umgebung zugeordnet und keiner Dienstinstanz in der Umgebung angehängt.

  Dies ist ein Übergangszustand für die Verlängerung der Lebensdauer einer Komponente über eine einzelne Dienstinstanz hinaus.

Die folgende Tabelle bietet einen Vergleich der verschiedenen Komponentenstatus auf oberster Ebene.

	Attached (Angefügt)	Detached (Getrennt)
Hauptzweck des Staates	Um die Infrastruktur einer Dienstinstanz zu erweitern.	Um die Infrastruktur der Komponente zwischen Anhängen der Serviceinstanz aufrechtzuerhalten.
Verbunden mit	Eine Dienstinstanz und eine Umgebung	Eine -Umgebung
Wichtige spezifische Eigenschaften	• Service-Name • Name der Dienstinstanz • Spezifikation	• Environment name
Kann gelöscht werden	✗ Nein	✓ Ja
Kann auf eine andere Dienstinstanz aktualisiert werden	✗ Nein	✓ Ja
Kann Eingaben lesen	✓ Ja	✗ Nein

Der Hauptzweck einer Komponente besteht darin, an eine Dienstinstanz angehängt zu werden und deren Infrastruktur um zusätzliche Ressourcen zu erweitern. Eine angehängte Komponente kann gemäß der Spezifikation Eingaben von der Dienstinstanz lesen. Sie können die Komponente nicht direkt löschen oder an eine andere Dienstinstanz anhängen. Sie können auch nicht die zugehörige

Dienstinstanz oder den zugehörigen Dienst und die zugehörige Umgebung löschen. Um eines dieser Dinge zu tun, aktualisieren Sie zuerst die Komponente, um sie von ihrer Dienstinstanz zu trennen.

Um die Infrastruktur der Komponente über die Lebensdauer einer einzelnen Dienstinstanz hinaus aufrechtzuerhalten, aktualisieren Sie die Komponente und trennen sie von ihrer Dienstinstanz, indem Sie die Namen des Dienstes und der Dienstinstanz entfernen. Bei diesem getrennten Zustand handelt es sich um einen Übergangszustand. Die Komponente hat keine Eingänge. Die Infrastruktur bleibt bereitgestellt und Sie können sie aktualisieren. Sie können Ressourcen löschen, denen die Komponente zugeordnet war, als sie angehängt wurde (Dienstinstanz, Dienst). Sie können die Komponente löschen oder sie so aktualisieren, dass sie erneut an eine Dienstinstanz angehängt wird.

Komponenteninfrastruktur als Codedateien

Component Infrastructure as Code (IaC) -Dateien ähneln denen für andere AWS Proton Ressourcen. Erfahren Sie hier mehr über einige Details, die für Komponenten spezifisch sind. Vollständige Informationen zum Erstellen von IaC-Dateien für finden Sie AWS Proton unter. Erstellung von Vorlagen und Bundles

Verwenden von Parametern mit Komponenten

Der AWS Proton Parameter-Namespace enthält einige Parameter, auf die eine Service-IaC-Datei verweisen kann, um den Namen und die Ausgaben einer zugehörigen Komponente abzurufen. Der Namespace enthält auch Parameter, auf die eine IaC-Komponenten-Datei verweisen kann, um Eingaben, Ausgaben und Ressourcenwerte aus der Umgebung, dem Dienst und der Dienstinstanz abzurufen, der die Komponente zugeordnet ist.

Eine Komponente hat keine eigenen Eingaben — sie bezieht ihre Eingaben von der Dienstinstanz, an die sie angehängt ist. Eine Komponente kann auch Umgebungsausgaben lesen.

Weitere Hinweise zur Verwendung von Parametern in IaC-Dateien für Komponenten und zugehörige Dienste finden Sie unterthe section called “ CloudFormation IaC-Parameter für Komponenten”. Allgemeine Informationen zu AWS Proton Parametern und eine vollständige Referenz zum Parameter-Namespace finden Sie unter. the section called “Parameters”

Robuste IaC-Dateien erstellen

Als Administrator können Sie beim Erstellen einer Service-Vorlagenversion entscheiden, ob Sie zulassen möchten, dass Dienstinstanzen, die anhand der Vorlagenversion erstellt wurden, Komponenten angehängt haben. Den supportedComponentSourcesParameter der

CreateServiceTemplateVersionAPI-Aktion finden Sie in der AWS Proton API-Referenz. Für jede future Dienstinstanz entscheidet jedoch die Person, die die Instanz erstellt, ob eine Komponente an sie angehängt wird oder nicht, und (im Fall von direkt definierten Komponenten) die Komponente IaC verfasst, ist in der Regel eine andere Person — ein Entwickler, der Ihre Dienstvorlage verwendet. Daher können Sie nicht garantieren, dass eine Komponente an eine Dienstinstanz angehängt wird. Sie können auch nicht garantieren, dass bestimmte Komponentenausgabenamen existieren oder dass die Werte dieser Ausgaben gültig und sicher sind.

AWS Proton und die Jinja-Syntax helfen Ihnen dabei, diese Probleme zu umgehen und robuste Dienstvorlagen zu erstellen, die auf folgende Weise fehlerfrei gerendert werden:

- AWS Proton Parameterfilter — Wenn Sie sich auf die Ausgabeeigenschaften von Komponenten beziehen, können Sie Parameterfilter verwenden. Dabei handelt es sich um Modifikatoren, die Parameterwerte validieren, filtern und formatieren. Weitere Informationen und Beispiele finden Sie unter the section called “CloudFormation Parameterfilter”.
- Standard mit einer einzigen Eigenschaft — Wenn Sie auf eine einzelne Ressource oder Ausgabeeigenschaft einer Komponente verweisen, können Sie sicherstellen, dass das Rendern Ihrer Service-Vorlage nicht fehlschlägt, indem Sie den `default` Filter mit oder ohne Standardwert verwenden. Wenn die Komponente oder ein bestimmter Ausgabeparameter, auf den Sie sich beziehen, nicht existiert, wird stattdessen der Standardwert (oder eine leere Zeichenfolge, falls Sie keinen Standardwert angegeben haben) gerendert, und das Rendern ist erfolgreich. Weitere Informationen finden Sie unter the section called “Geben Sie Standardwerte an”.

  Beispiele:

  - `{{ service_instance.components.default.name | default("") }}`
  - `{{ service_instance.components.default.outputs.my-output | default("17") }}`

  > **Note**
  >
  > Verwechseln Sie den `.default` Teil des Namespaces, der direkt definierte Komponenten bezeichnet, nicht mit dem `default` Filter, der einen Standardwert bereitstellt, wenn die referenzierte Eigenschaft nicht existiert.

- Gesamter Objektverweis — Wenn Sie auf die gesamte Komponente oder auf die Auflistung der Ausgaben einer Komponente verweisen, wird ein leeres Objekt AWS Proton zurückgegeben und somit garantiert`{}`, dass das Rendern Ihrer Dienstvorlage nicht fehlschlägt. Sie müssen keinen

Filter verwenden. Stellen Sie sicher, dass Sie die Referenz in einem Kontext erstellen, der ein leeres Objekt annehmen kann, oder verwenden Sie eine `{{ if .. }}` Bedingung, um zu testen, ob ein leeres Objekt vorhanden ist.

Beispiele:

- `{{ service_instance.components.default }}`
- `{{ service_instance.components.default.outputs }}`

CloudFormation Beispiel für eine Komponente

Hier ist ein vollständiges Beispiel für eine AWS Proton direkt definierte Komponente und wie Sie sie in einem AWS Proton Dienst verwenden können. Die Komponente stellt einen Amazon Simple Storage Service (Amazon S3) -Bucket und die zugehörige Zugriffsrichtlinie bereit. Die Service-Instance kann auf diesen Bucket verweisen und ihn verwenden. Der Bucket-Name basiert auf den Namen der Umgebung, des Dienstes, der Dienstinstanz und der Komponente, was bedeutet, dass der Bucket mit einer bestimmten Instanz der Komponentenvorlage verknüpft ist, die eine bestimmte Dienstinstanz erweitert. Entwickler können auf der Grundlage dieser Komponentenvorlage mehrere Komponenten erstellen, um Amazon S3 S3-Buckets für unterschiedliche Service-Instances und funktionale Anforderungen bereitzustellen.

Das Beispiel behandelt das Verfassen der verschiedenen erforderlichen IaC-Dateien (CloudFormation Infrastructure as Code) und das Erstellen einer erforderlichen Rolle AWS Identity and Access Management (IAM). Das Beispiel gruppiert die Schritte nach den Rollen, denen sie gehören.

Schritte des Administrators

Um Entwicklern die Verwendung von Komponenten mit einem Dienst zu ermöglichen

1. Erstellen Sie eine AWS Identity and Access Management (IAM-) Rolle, die den Umfang der Ressourcen abgrenzt, die direkt definierte Komponenten, die in Ihrer Umgebung ausgeführt werden, bereitstellen können. AWS Proton übernimmt diese Rolle später, um direkt definierte Komponenten in der Umgebung bereitzustellen.

   Verwenden Sie für dieses Beispiel die folgende Richtlinie:

Example direkt definierte Komponentenrolle

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. Geben Sie die Rolle an, die Sie im vorherigen Schritt erstellt haben, wenn Sie die Umgebung erstellen oder aktualisieren. Geben Sie in der AWS Proton Konsole auf der Seite „Umgebung konfigurieren“ eine Komponentenrolle an. Wenn Sie die AWS Proton API oder verwenden AWS CLI, geben Sie die `componentRoleArn` der CreateEnvironmentoder UpdateEnvironmentAPI-Aktionen an.
3. Erstellen Sie eine Dienstvorlage, die sich auf eine direkt definierte Komponente bezieht, die an die Dienstinstanz angehängt ist.

   Das Beispiel zeigt, wie eine robuste Dienstvorlage geschrieben wird, die nicht kaputt geht, wenn keine Komponente an die Dienstinstanz angehängt ist.

   Example CloudFormation Service-IaC-Datei mithilfe einer Komponente

```
# 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. Erstellen Sie eine neue Nebenversion der Dienstvorlage, in der direkt definierte Komponenten als unterstützt deklariert werden.

   - Vorlagenpaket in Amazon S3 — Wenn Sie in der AWS Proton Konsole eine Service-Vorlagenversion erstellen, wählen Sie für Unterstützte Komponentenquellen die Option Direkt definiert aus. Wenn Sie die AWS Proton API oder verwenden AWS CLI, geben Sie `DIRECTLY_DEFINED` im `supportedComponentSources` Parameter CreateServiceTemplateVersionoder UpdateServiceTemplateVersionAPI-Aktionen an.
   - Vorlagensynchronisierung — Übernehmen Sie eine Änderung an Ihrem Service-Template-Bundle-Repository, das Sie `DIRECTLY_DEFINED` als Element `supported_component_sources:` in der `.template-registration.yaml` Datei im Hauptversionsverzeichnis angeben. Weitere Informationen über diese Datei finden Sie unter the section called “Synchronisieren von Dienstvorlagen”.

5. Veröffentlichen Sie die neue Nebenversion der Service-Vorlage. Weitere Informationen finden Sie unter the section called “Veröffentlichen”.
6. Stellen Sie sicher, dass Sie Entwicklern, die diese Dienstvorlage verwenden, die Rolle `proton:CreateComponent` in der IAM-Rolle zuweisen.

Schritte für Entwickler

Um eine direkt definierte Komponente mit einer Dienstinstanz zu verwenden

1. Erstellen Sie einen Dienst, der die Version der Dienstvorlage verwendet, die der Administrator mit Komponentenunterstützung erstellt hat. Sie können auch eine Ihrer vorhandenen Dienstinstanzen aktualisieren, um die neueste Vorlagenversion zu verwenden.

2. Schreiben Sie eine IaC-Vorlagendatei für Komponenten, die einen Amazon S3 S3-Bucket und eine zugehörige Zugriffsrichtlinie bereitstellt und diese Ressourcen als Ausgaben bereitstellt.

   Example Komponenten-IaC-Datei CloudFormation

```
# 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. Wenn Sie die AWS Proton API oder verwenden AWS CLI, schreiben Sie eine Manifestdatei für die Komponente.

   Example direkt definiertes Komponentenmanifest

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

```
    template_language: cloudformation
```

4. Erstellen Sie eine direkt definierte Komponente. AWS Proton nimmt die Komponentenrolle an, die der Administrator für die Bereitstellung der Komponente definiert hat.

   Wählen Sie in der AWS Proton Konsole auf der Seite Komponenten die Option Komponente erstellen aus. Geben Sie unter Komponenteneinstellungen einen Komponentennamen und optional eine Komponentenbeschreibung ein. Wählen Sie für Komponentenanhang die Option Komponente an eine Dienstinstanz anhängen aus. Wählen Sie Ihre Umgebung, Ihren Service und Ihre Dienstinstanz aus. Wählen Sie CloudFormationals Komponentenquelle die Option und anschließend die Komponenten-IaC-Datei aus.

   > **Note**
   >
   > Sie müssen kein Manifest angeben — die Konsole erstellt eines für Sie.

   Wenn Sie die AWS Proton API oder verwenden AWS CLI, verwenden Sie die CreateComponentAPI-Aktion. Legen Sie eine Komponente fest `name` und sind optional`description`. Stellen Sie `environmentName``serviceName`, und ein`serviceInstanceName`. Legen Sie `templateSource` `manifest` die Pfade der von Ihnen erstellten Dateien fest.

   > **Note**
   >
   > Die Angabe eines Umgebungsnamens ist optional, wenn Sie Dienst- und Dienstinstanznamen angeben. Die Kombination dieser beiden ist in Ihrem AWS Konto einzigartig und AWS Proton kann die Umgebung anhand der Dienstinstanz bestimmen.

5. Aktualisieren Sie Ihre Dienstinstanz, um sie erneut bereitzustellen. AWS Proton verwendet Ausgaben Ihrer Komponente in der gerenderten Service-Instance-Vorlage, damit Ihre Anwendung den Amazon S3 S3-Bucket verwenden kann, den die Komponente bereitgestellt hat.

Git-Repositorys verwenden mit AWS Proton

AWS Proton verwendet Git-Repositorys für eine Vielzahl von Zwecken. In der folgenden Liste werden die mit AWS Proton Ressourcen verknüpften Repository-Typen kategorisiert. Für AWS Proton Funktionen, die wiederholt eine Verbindung zu Ihrem Repository herstellen, um entweder Inhalte dorthin zu übertragen oder Inhalte daraus abzurufen, müssen Sie AWS Proton in Ihrem AWS Konto einen Repository-Link registrieren. Ein Repository-Link besteht aus einer Reihe von Eigenschaften, die verwendet AWS Proton werden können, wenn eine Verbindung zu einem Repository hergestellt wird. AWS Proton unterstützt GitHub derzeit GitHub Enterprise und BitBucket.

Entwickler-Repositorien

Code-Repository — Ein Repository, das Entwickler zum Speichern von Anwendungscode verwenden. Wird für die Codebereitstellung verwendet. AWS Proton interagiert nicht direkt mit diesem Repository. Wenn ein Entwickler einen Dienst bereitstellt, der eine Pipeline enthält, gibt er den Namen und den Zweig des Repositorys an, aus dem der Anwendungscode gelesen werden soll. AWS Proton leitet diese Informationen an die Pipeline weiter, die er bereitstellt.

Weitere Informationen finden Sie unter the section called “Create”.

Administrator-Repositorien

Vorlagen-Repository — Ein Repository, in dem Administratoren AWS Proton Vorlagenpakete speichern. Wird für die Vorlagensynchronisierung verwendet. Wenn ein Administrator eine Vorlage in erstellt AWS Proton, kann er auf ein Vorlagen-Repository verweisen und die neue Vorlage damit synchron AWS Proton halten. Wenn der Administrator das Vorlagenpaket im Repository aktualisiert, AWS Proton wird automatisch eine neue Vorlagenversion erstellt. Verknüpfen Sie ein Vorlagen-Repository mit, AWS Proton bevor Sie es für die Synchronisierung verwenden können.

Weitere Informationen finden Sie unter the section called “Konfigurationen für die Vorlagensynchronisierung”.

> **Note**
>
> Ein Vorlagen-Repository ist nicht erforderlich, wenn Sie Ihre Vorlagen weiterhin auf Amazon Simple Storage Service (Amazon S3) hochladen und die AWS Proton Vorlagenverwaltung aufrufen, APIs um neue Vorlagen oder Vorlagenversionen zu erstellen.

Selbstverwaltete Bereitstellungs-Repositorys

Infrastruktur-Repository — Ein Repository, das gerenderte Infrastrukturvorlagen hostet. Wird für die selbstverwaltete Bereitstellung der Ressourceninfrastruktur verwendet. Wenn ein Administrator eine Umgebung für die selbstverwaltete Bereitstellung erstellt, stellt er ein Repository bereit. AWS Proton sendet Pull-Requests (PRs) an dieses Repository, um die Infrastruktur für die Umgebung und für alle in der Umgebung bereitgestellten Dienstinstanzen zu erstellen. Verknüpfen Sie ein Infrastruktur-Repository mit, AWS Proton bevor Sie es für die selbstverwaltete Infrastrukturbereitstellung verwenden können.

Pipeline-Repository — Ein Repository, das zum Erstellen von Pipelines verwendet wird. Wird für die selbstverwaltete Bereitstellung von Pipelines verwendet. Die Verwendung eines zusätzlichen Repositorys zur Bereitstellung von Pipelines ermöglicht AWS Proton das Speichern von Pipeline-Konfigurationen unabhängig von einzelnen Umgebungen oder Diensten. Sie müssen nur ein einziges Pipeline-Repository für all Ihre selbstverwalteten Bereitstellungsdienste bereitstellen. Verknüpfen Sie ein Pipeline-Repository mit, AWS Proton bevor Sie es für die selbstverwaltete Pipeline-Bereitstellung verwenden können.

Weitere Informationen finden Sie unter the section called “AWS-verwaltete Bereitstellung”.

Themen

Erstellen Sie einen Link zu Ihrem Repository

Sie können mit der Konsole oder der CLI einen Link zu Ihrem Repository erstellen. Wenn Sie einen Repository-Link erstellen, AWS Proton wird für Sie eine serviceverknüpfte Rolle erstellt.

AWS-Managementkonsole

Erstellen Sie einen Link zu Ihrem Repository, wie in den folgenden Konsolenschritten gezeigt.

1. Wählen Sie in der AWS Proton Konsole Repositories aus.
2. Wählen Sie Repository erstellen aus.

3. Gehen Sie auf der Seite Neues Repository verknüpfen im Abschnitt Repository-Details wie folgt vor:

   a. Wählen Sie Ihren Repository-Anbieter.

   b. Wählen Sie eine Ihrer bestehenden Verbindungen. Falls Sie noch keine haben, wählen Sie Neue CodeStar Verbindung hinzufügen aus, um eine Verbindung herzustellen. Kehren Sie dann zur AWS Proton Konsole zurück, aktualisieren Sie die Verbindungsliste und wählen Sie Ihre neue Verbindung aus.

   c. Wählen Sie aus Ihren verbundenen Quellcode-Repositorys.

4. [optional] Wählen Sie im Abschnitt „Tags“ die Option „Neues Tag hinzufügen“ aus und geben Sie Schlüssel - und Wertepaare ein.

5. Wählen Sie Repository erstellen aus.

6. Sehen Sie sich die Detaildaten für Ihr verknüpftes Repository an.

AWS CLI

Erstellen und registrieren Sie einen Link zu Ihrem Repository.

Führen Sie den folgenden Befehl aus:

```
$ 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
```

Die letzten beiden Parameter --encryption-key und--tags, sind optional.

Antwort:

```
{
    "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"
    }
}
```

Nachdem Sie einen Repository-Link erstellt haben, können Sie eine Liste mit AWS und vom Kunden verwalteten Tags anzeigen, wie im folgenden Beispielbefehl gezeigt. AWS Proton generiert automatisch AWS verwaltete Tags für Sie. Mit dem können Sie auch vom Kunden verwaltete Tags ändern und erstellen AWS CLI. Weitere Informationen finden Sie unter AWS Proton Ressourcen und Tagging.

Befehl:

```
$ aws proton list-tags-for-resource \
    --resource-arn "arn:aws:proton:region-id:123456789012:repository/github:myrepos/
environments"
```

Verknüpfte Repository-Daten anzeigen

Sie können die Details des verknüpften Repositorys mithilfe der Konsole oder der auflisten und anzeigen AWS CLI. Für Repository-Links, mit denen Git-Repositorys synchronisiert werden AWS Proton, kannst du die Definition und den Status der Repository-Synchronisierung mit dem AWS CLI abrufen.

AWS-Managementkonsole

Mithilfe der AWS Proton Konsole können Sie die Details des verknüpften Repositorys auflisten und anzeigen.

1. Um eine Liste Ihrer verknüpften Repositorys anzuzeigen, wählen Sie im Navigationsbereich Repositorys aus.
2. Um Detaildaten anzuzeigen, wählen Sie den Namen eines Repositorys.

AWS CLI

Listet Ihre verknüpften Repositorys auf.

Führen Sie den folgenden Befehl aus:

```
$ aws proton list-repositories
```

Antwort:

```
{
    "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"
        }
    ]
}
```

Sehen Sie sich die Details eines verknüpften Repositorys an.

Führen Sie den folgenden Befehl aus:

```
$ aws proton get-repository \
    --name myrepos/templates \
    --provider "GITHUB"
```

Antwort:

```
{
    "repository": {
        "arn": "arn:aws:proton:region-id:123456789012:repository/github:myrepos/
templates",
        "name": "myrepos/templates",
        "provider": "GITHUB"
    }
}
```

Listet Ihre synchronisierten Repositorys auf.

Das folgende Beispiel listet Repositorys auf, die Sie für die Vorlagensynchronisierung konfiguriert haben.

Führen Sie den folgenden Befehl aus:

```
$ aws proton list-repository-sync-definitions \
    --branch "main" \
    --repository-name myrepos/templates \
    --repository-provider "GITHUB" \
    --sync-type "TEMPLATE_SYNC"
```

Status der Repository-Synchronisierung anzeigen.

Im folgenden Beispiel wird der Synchronisierungsstatus eines Template-Sync-Repositorys abgerufen.

Führen Sie den folgenden Befehl aus:

```
$ aws proton get-repository-sync-status \
    --branch "main" \
    --repository-name myrepos/templates \
    --repository-provider "GITHUB" \
    --sync-type "TEMPLATE_SYNC"
```

Antwort:

```
{
    "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"
    }
}
```

Einen Repository-Link löschen

Sie können einen Repository-Link mithilfe der Konsole oder der AWS CLI löschen.

> **Note**
>
> Durch das Löschen eines Repository-Links wird nur der registrierte Link entfernt, der AWS Proton sich in Ihrem AWS Konto befindet. Es werden keine Informationen aus Ihrem Repository gelöscht.

AWS-Managementkonsole

Löschen Sie einen Repository-Link mithilfe der Konsole.

Auf der Repository-Detailseite.

1. Wählen Sie in der AWS Proton Konsole Repositories aus.
2. Wählen Sie in der Liste der Repositorys das Optionsfeld links neben dem Repository aus, das Sie löschen möchten.
3. Wählen Sie Löschen aus.
4. In einem Modal werden Sie aufgefordert, die Aktion Löschen zu bestätigen.
5. Folgen Sie den Anweisungen und wählen Sie Ja, löschen.

AWS CLI

Löschen Sie einen Repository-Link.

Führen Sie den folgenden Befehl aus:

```
$ aws proton delete-repository \
    --name myrepos/templates \
    --provider"GITHUB"
```

Antwort:

```
{
    "repository": {
        "arn": "arn:aws:proton:region-id:123456789012:repository/github:myrepos/
templates",
        "name": "myrepos/templates",
        "provider": "GITHUB"
    }
}
```

Überwachung AWS Proton

Die Überwachung ist ein wichtiger Bestandteil der Aufrechterhaltung der Zuverlässigkeit, Verfügbarkeit AWS Proton und Leistung Ihrer AWS Lösungen. Im folgenden Abschnitt werden Überwachungstools beschrieben, die Sie zusammen verwenden können AWS Proton.

Automatisieren Sie AWS Proton mit EventBridge

Sie können AWS Proton Ereignisse in Amazon überwachen EventBridge. EventBridge liefert einen Stream von Echtzeitdaten aus Ihren eigenen Anwendungen, software-as-a-service (SaaS-) Anwendungen und AWS-Services. Sie können Ereignisse so konfigurieren, dass sie auf Änderungen des AWS Ressourcenstatus reagieren. EventBridge leitet diese Daten dann an Zieldienste wie AWS Lambda Amazon Simple Notification Service weiter. Diese Ereignisse entsprechen denen, die in Amazon CloudWatch Events erscheinen. CloudWatch Events bietet einen Stream von Systemereignissen, die Änderungen an AWS Ressourcen beschreiben, nahezu in Echtzeit. Weitere Informationen finden Sie unter Was ist Amazon EventBridge? im EventBridge Amazon-Benutzerhandbuch.

Wird verwendet EventBridge , um über Statusänderungen in den AWS Proton Bereitstellungs-Workflows informiert zu werden.

Event types (Ereignistypen)

Ereignisse bestehen aus Regeln, die ein Ereignismuster und Ziele enthalten. Sie konfigurieren eine Regel, indem Sie ein Ereignismuster und Zielobjekte auswählen:

Ereignismuster

> Jede Regel wird als ein Ereignismuster mit der Quelle und Art der zu überwachenden Ereignisse sowie den Ereigniszielen ausgedrückt. Um Ereignisse zu überwachen, erstellen Sie eine Regel mit dem Dienst, den Sie überwachen, als Ereignisquelle. Sie können beispielsweise eine Regel mit einem Ereignismuster erstellen, das AWS Proton als Ereignisquelle verwendet wird, um eine Regel auszulösen, wenn sich ein Bereitstellungsstatus ändert.

Ziele

> Die Regel empfängt einen ausgewählten Dienst als Ereignisziel. Sie können einen Zieldienst einrichten, um Benachrichtigungen zu senden, Statusinformationen zu erfassen, Abhilfemaßnahmen zu ergreifen, Ereignisse einzuleiten oder andere Aktionen zu ergreifen.

Ereignisobjekte enthalten Standardfelder für ID, Konto AWS-Region, Detailtyp, Quelle, Version, Ressource und Zeit (optional). Das Detailfeld ist ein verschachteltes Objekt, das benutzerdefinierte Felder für das Ereignis enthält.

AWS Proton Ereignisse werden nach bestem Wissen und Gewissen ausgegeben. Bereitstellung nach bestem Wissen bedeutet, dass der Service versucht, alle Ereignisse an zu senden EventBridge, aber in einigen seltenen Fällen kann es vorkommen, dass ein Ereignis nicht zugestellt wird.

In der folgenden Tabelle sind für jede AWS Proton Ressource, die Ereignisse auslösen kann, der Detailwert, die Detailfelder und (sofern verfügbar) ein Verweis auf eine Werteliste für die Felder `status` und `previousStatus` Detail aufgeführt. Wenn eine Ressource gelöscht wird, lautet der Wert des `status` Detailfeldes. `DELETED`

Ressource	Wert vom Typ Detail	Felder mit Details

`EnvironmentTemplate`	AWS Proton Änderung des Status der Umgebungsvorlage	`name`
		`status`
		`previousStatus`

`EnvironmentTemplateVersion`	AWS Proton Änderung des Versionsstatus der Umgebungsvorlage	`name`
		`majorVersion`
		`minorVersion`
		`status`
		`previousStatus`
		Statuswerte

`ServiceTemplate`	AWS Proton Änderung des Status der Dienstvorlage	`name` `status`

Ressource	Wert vom Typ Detail	Felder mit Details
		`previousStatus`

`ServiceTemplateVersion`	AWS Proton Änderung des Versionsstatus der Dienstvorlage	`name`
		`majorVersion`
		`minorVersion`
		`status`
		`previousStatus`
		Statuswerte

`Environment`	AWS Proton Änderung des Umgebungsstatus	`name`
		`status`
		`previousStatus`

`Service`	AWS Proton Änderung des Servicestatus	`name`
		`status`
		`previousStatus`
		Statuswerte

Ressource	Wert vom Typ Detail	Felder mit Details

`ServiceInstance`	AWS Proton Änderung des Status der Dienstinstanz	`name`
		`serviceName`
		`status`
		`previousStatus`

`ServicePipeline`	AWS Proton Änderung des Status der Service-Pipeline	`serviceName`
		`status`
		`previousStatus`

`EnvironmentAccountConnection`	AWS Proton Änderung des Verbindungsstatus des Umgebungskontos	`id`
		`status`
		`previousStatus`
		Statuswerte

`Component`	AWS Proton Änderung des Komponentenstatus	`name`
		`status`
		`previousStatus`

AWS Proton Beispiele für Ereignisse

Die folgenden Beispiele zeigen, wie Ereignisse an gesendet AWS Proton werden können EventBridge.

Service-Vorlage

```
{
    "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"
    }
}
```

Version der Dienstvorlage

```
{
    "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"
    }
}
```

Umgebung

```
{
    "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: Senden Sie Amazon Simple Notification Service-Benachrichtigungen über Änderungen des AWS Proton Servicestatus

In diesem Tutorial verwenden Sie eine AWS Proton vorkonfigurierte Ereignisregel, die Statusänderungen für Ihren AWS Proton Service erfasst. EventBridgesendet die Statusänderungen an ein Amazon SNS SNS-Thema. Sie abonnieren das Thema und Amazon SNS sendet Ihnen E-Mails zur Statusänderung für Ihren AWS Proton Service.

Voraussetzungen

Sie haben einen bestehenden AWS Proton Service mit einem `Active` Status. Im Rahmen dieses Tutorials können Sie diesem Service Dienstinstanzen hinzufügen und die Instanzen anschließend löschen.

Wenn Sie einen AWS Proton Dienst erstellen müssen, finden Sie weitere Informationen unterErste Schritte. Weitere Informationen erhalten Sie unter AWS Proton Kontingente und the section called “Edit (Bearbeiten)”.

Schritt 1: Erstellen und Abonnieren eines Amazon-SNS-Themas

Erstellen Sie ein Amazon SNS SNS-Thema, das als Ereignisziel für die in Schritt 2 erstellte Ereignisregel dient.

Erstellen Sie ein Amazon SNS-Thema.

1. Melden Sie sich an und öffnen Sie die Amazon SNS SNS-Konsole.
2. Wählen Sie im Navigationsbereich Themen, Thema erstellen aus.
3. Gehen Sie auf der Seite Thema erstellen wie folgt vor:

    a. Wählen Sie Typ Standard.

    b. Geben Sie als Namen Thema ein **`tutorial-service-status-change`** und wählen Sie Create topic aus.

4. Wählen Sie auf der tutorial-service-status-changeDetailseite die Option Abonnement erstellen aus.
5. Gehen Sie auf der Seite Abonnement erstellen wie folgt vor:
   a. Wählen Sie unter Protocol (Protokoll) die Option Email (E-Mail) aus.
   b. Geben Sie für Endpunkt eine E-Mail-Adresse ein, auf die Sie aktuell Zugriff haben, und wählen Sie Abonnement erstellen aus.
6. Überprüfen Sie Ihr E-Mail-Konto und warten Sie auf eine E-Mail-Nachricht zur Bestätigung Ihres Abonnements. Wenn Sie es erhalten, öffnen Sie es und wählen Sie Abonnement bestätigen.

Schritt 2: Registrieren von Ereignisregeln

Registrieren Sie eine Ereignisregel, die Statusänderungen für Ihren AWS Proton Service erfasst. Weitere Informationen finden Sie unter Voraussetzungen.

Erstellen Sie eine Ereignisregel.

1. Öffnen Sie die EventBridge Amazon-Konsole.
2. Wählen Sie im Navigationsbereich Events (Ereignisse) und Rules (Regeln) aus.
3. Wählen Sie auf der Seite Regeln im Abschnitt Regeln die Option Regel erstellen aus.
4. Gehen Sie auf der Seite Regel erstellen wie folgt vor:
   a. Geben Sie im Abschnitt Name und Beschreibung für Name den Wert ein**`tutorial-rule`**.
   b. Wählen Sie im Abschnitt Muster definieren die Option Ereignismuster aus.
      i. Wählen Sie unter Event matching pattern (Ereignisübereinstimmungsmuster) die Option Pre-defined by service (Vordefiniertes Muster nach Service) aus.
      ii. Wählen Sie für Service provider (Serviceanbieter) die Option AWS aus.
      iii. Wählen Sie für Service name (Servicename) AWS Proton aus.
      iv. Wählen Sie als Ereignistyp die Option AWS Proton Servicestatusänderung aus.

      Das Ereignismuster wird in einem Texteditor angezeigt.
      v. Öffnen Sie die AWS Proton -Konsole.
      vi. Wählen Sie im Navigationsbereich Services.
      vii. Wählen Sie auf der Seite Dienste den Namen Ihres AWS Proton Dienstes aus.

viii. Kopieren Sie auf der Seite mit den Servicedetails den Service Amazon Resource Name (ARN).

ix. Gehen Sie zurück zur EventBridge Konsole und zu Ihrer Tutorial-Regel und wählen Sie im Texteditor Bearbeiten aus.

x. Geben Sie im Texteditor für `"resources":` den Dienst-ARN ein, den Sie in Schritt viii kopiert haben.

```
{
    "source": ["aws.proton"],
    "detail-type": ["AWS Proton Service Status Change"],
    "resources": ["arn:aws:proton:region-id:123456789012:service/your-service"]
}
```

xi. Speichern Sie das Ereignismuster.

c. Gehen Sie im Abschnitt Ziele auswählen wie folgt vor:

i. Wählen Sie in Target (Ziel) die Option SNS topic (SNS-Thema) aus.

ii. Wählen Sie unter Thema die Option tutorial-service-status-change.

d. Wählen Sie Erstellen aus.

Schritt 3: Testen Sie Ihre Eventregel

Stellen Sie sicher, dass Ihre Ereignisregel funktioniert, indem Sie Ihrem AWS Proton Service eine Instanz hinzufügen.

1. Wechseln Sie zur AWS Proton Konsole.
2. Wählen Sie im Navigationsbereich Services.
3. Wählen Sie auf der Seite Dienste den Namen Ihres Dienstes aus.
4. Wählen Sie auf der Seite mit den Dienstdetails die Option Bearbeiten aus.
5. Wählen Sie auf der Seite Service konfigurieren die Option Weiter aus.
6. Wählen Sie auf der Seite Benutzerdefinierte Einstellungen konfigurieren im Abschnitt Dienstinstanzen die Option Neue Instanz hinzufügen aus.
7. Füllen Sie das Formular für Ihre neue Instanz aus:

a. Geben Sie einen Namen für Ihre neue Instanz ein.

b. Wählen Sie dieselben kompatiblen Umgebungen aus, die Sie für Ihre vorhandenen Instances ausgewählt haben.

c. Geben Sie Werte für die erforderlichen Eingaben ein.

d. Wählen Sie Weiter aus.

8. Überprüfen Sie Ihre Eingaben und wählen Sie Aktualisieren.

9. Wenn der Dienststatus angezeigt wird`Active`, überprüfen Sie Ihre E-Mails, um sicherzustellen, dass Sie AWS Benachrichtigungen mit Statusaktualisierungen erhalten haben.

```
{
    "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"
    }
}
```

Schritt 4: Bereinigen

Löschen Sie Ihr Amazon SNS SNS-Thema und Ihr Abonnement und löschen Sie Ihre EventBridge Regel.

Löschen Sie Ihr Amazon SNS SNS-Thema und Ihr Abonnement.

1. Navigieren Sie zur Amazon SNS SNS-Konsole.
2. Wählen Sie im Navigationsbereich Subscriptions (Abonnements) aus.
3. Wählen Sie auf der Seite Abonnements das Abonnement aus, das Sie für das angegebene Thema abgeschlossen haben, `tutorial-service-status-change` und klicken Sie dann auf Löschen.
4. Wählen Sie im Navigationsbereich Themen aus.
5. Wählen Sie auf der Themenseite das angegebene Thema aus `tutorial-service-status-change` und klicken Sie dann auf Löschen.
6. In einem Modalfenster werden Sie aufgefordert, den Löschvorgang zu überprüfen. Folgen Sie den Anweisungen und wählen Sie Löschen.

Lösche deine EventBridge Regel.

1. Navigieren Sie zur EventBridge Amazon-Konsole.
2. Wählen Sie im Navigationsbereich Events (Ereignisse) und Rules (Regeln) aus.
3. Wählen Sie auf der Seite „Regeln“ die angegebene Regel aus `tutorial-rule` und klicken Sie dann auf Löschen.
4. In einem Modalfenster werden Sie aufgefordert, den Löschvorgang zu überprüfen. Wählen Sie Löschen aus.

Löschen Sie die hinzugefügte Dienstinstanz.

1. Navigieren Sie zur AWS Proton -Konsole.
2. Wählen Sie im Navigationsbereich Services.
3. Wählen Sie auf der Seite Dienste den Namen Ihres Dienstes aus.
4. Wählen Sie auf der Seite mit den Dienstdetails Bearbeiten und dann Weiter aus.

5. Wählen Sie auf der Seite Benutzerdefinierte Einstellungen konfigurieren im Abschnitt Dienstinstanzen die Option Löschen für die Dienstinstanz aus, die Sie im Rahmen dieses Tutorials erstellt haben, und klicken Sie dann auf Weiter.
6. Überprüfen Sie Ihre Eingaben und wählen Sie Aktualisieren.
7. In einem Modal werden Sie aufgefordert, den Löschvorgang zu überprüfen. Folgen Sie den Anweisungen und wählen Sie Ja, löschen.

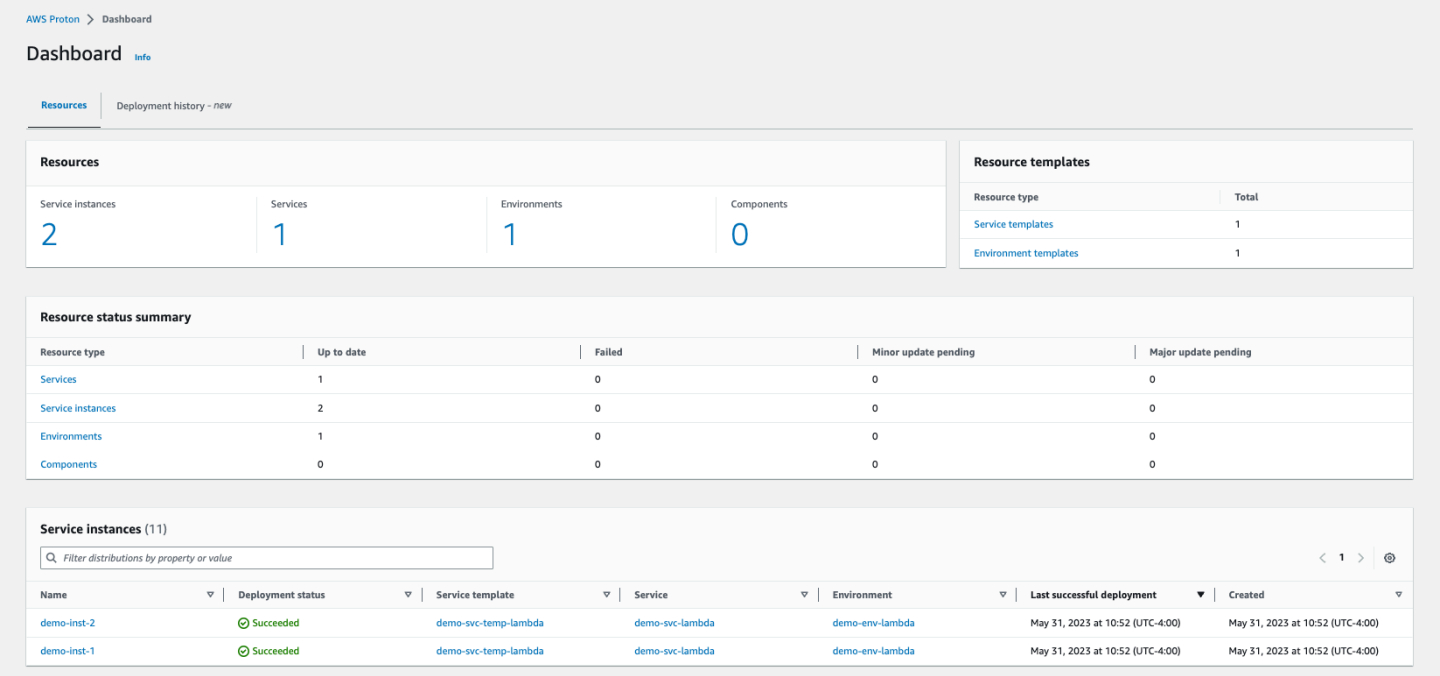
Halten Sie die Infrastruktur mit dem AWS Proton Dashboard auf dem neuesten Stand

Das AWS Proton Dashboard bietet eine Zusammenfassung der AWS Proton Ressourcen in Ihrem AWS Konto, wobei ein besonderer Schwerpunkt auf der Veralterung liegt, d. h. darauf, wie aktuell die bereitgestellten Ressourcen sind. Eine bereitgestellte Ressource ist aktuell, wenn sie die empfohlene Version der zugehörigen Vorlage verwendet. Für eine out-of-date bereitgestellte Ressource ist möglicherweise eine größere oder kleinere Aktualisierung der Vorlagenversion erforderlich.

Sehen Sie sich das Dashboard in der AWS Proton Konsole an

Um das AWS Proton Dashboard anzuzeigen, öffnen Sie die AWS Proton Konsole und wählen Sie dann im Navigationsbereich Dashboard aus.

Ressourcen

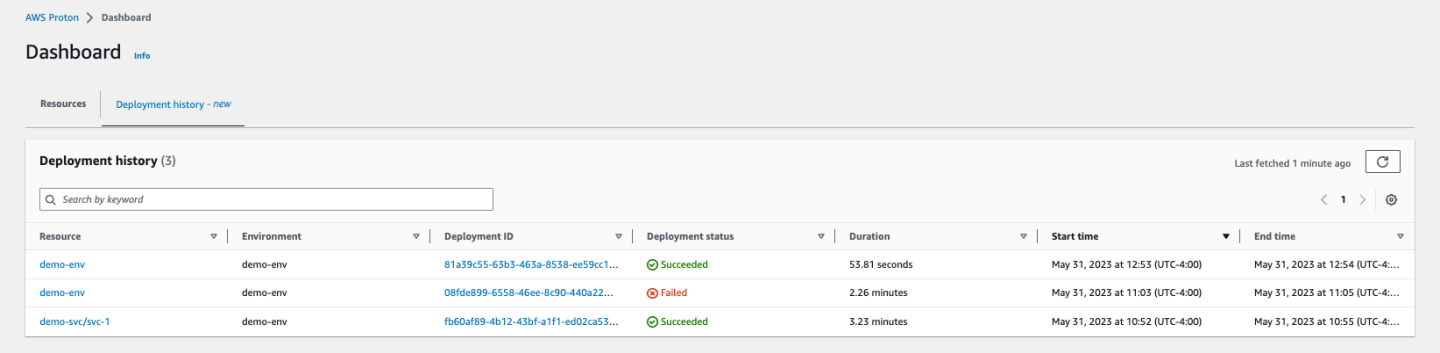


Auf der ersten Registerkarte des Dashboards wird die Anzahl aller Ressourcen in Ihrem Konto angezeigt. Auf der Registerkarte Ressourcen werden die Anzahl Ihrer Dienstinstanzen, Dienste, Umgebungen und Komponenten sowie Ihre Ressourcenvorlagen angezeigt. Außerdem wird die Anzahl der Ressourcen für jeden bereitgestellten Ressourcentyp nach dem Status der Ressourcen dieses Typs aufgeschlüsselt. Eine Service-Instanztabelle enthält Details zu jeder Service-Instanz — ihren Bereitstellungsstatus, die AWS Proton Ressourcen, denen sie zugeordnet ist, die Updates, die für sie verfügbar sind, und einige Zeitstempel.

Sie können die Liste der Serviceinstanzen nach einer beliebigen Tabelleneigenschaft filtern. Sie können beispielsweise nach Serviceinstanzen filtern, die innerhalb eines bestimmten Zeitfensters bereitgestellt wurden, oder nach Serviceinstanzen, die im Vergleich zu den Empfehlungen für Haupt- oder Nebenversionen veraltet sind.

Wählen Sie einen Namen für die Serviceinstanz, um zur Detailseite der Serviceinstanz zu gelangen, auf der Sie entsprechende Versionsupdates vornehmen können. Wählen Sie einen anderen AWS Proton Ressourcennamen, um zur zugehörigen Detailseite zu navigieren, oder wählen Sie einen Ressourcentyp, um zur entsprechenden Ressourcenliste zu navigieren.

Bereitstellungsverlauf



Auf der Registerkarte „Bereitstellungsverlauf“ können Sie Details zu Ihren Bereitstellungen einsehen. In der Tabelle mit dem Bereitstellungsverlauf können Sie den Bereitstellungsstatus sowie die Umgebung und die Bereitstellungs-ID verfolgen. Sie können den Ressourcennamen oder die Bereitstellungs-ID wählen, um noch mehr Details zu sehen, z. B. eine Meldung zum Bereitstellungsstatus und Ressourcenausgaben. In der Tabelle können Sie auch nach einer beliebigen Tabelleneigenschaft filtern.

Sicherheit in AWS Proton

Cloud-Sicherheit AWS hat höchste Priorität. Als AWS Kunde profitieren Sie von Rechenzentren und Netzwerkarchitekturen, die darauf ausgelegt sind, die Anforderungen der sicherheitssensibelsten Unternehmen zu erfüllen.

Sicherheit ist eine gemeinsame AWS Verantwortung von Ihnen und Ihnen. Das Modell der geteilten Verantwortung beschreibt dies als Sicherheit der Cloud und Sicherheit in der Cloud:

- Sicherheit der Cloud — AWS ist verantwortlich für den Schutz der Infrastruktur, die AWS-Services in der läuft AWS Cloud. AWS bietet Ihnen auch Dienste, die Sie sicher nutzen können. Externe Prüfer testen und verifizieren regelmäßig die Wirksamkeit unserer Sicherheitsmaßnahmen im Rahmen der AWS . Weitere Informationen zu den Compliance-Programmen, die für gelten AWS Proton, finden Sie AWS-Services unter Umfang nach Compliance-ProgrammenAWS-Services und unter .
- Sicherheit in der Cloud — Ihre Verantwortung richtet sich nach dem AWS-Service , was Sie verwenden. Sie sind auch für andere Faktoren verantwortlich, etwa für die Vertraulichkeit Ihrer Daten, für die Anforderungen Ihres Unternehmens und für die geltenden Gesetze und Vorschriften.

Diese Dokumentation hilft Ihnen zu verstehen, wie Sie das Modell der gemeinsamen Verantwortung bei der Verwendung anwenden können AWS Proton. Die folgenden Themen veranschaulichen, wie Sie AWS Proton zur Erfüllung Ihrer Sicherheits- und Compliance-Ziele konfigurieren können. Sie erfahren auch, wie Sie andere verwenden können AWS-Services , die Ihnen bei der Überwachung und Sicherung Ihrer AWS Proton Ressourcen helfen.

Topics

Identity and Access Management für AWS Proton

AWS Identity and Access Management (IAM) hilft einem Administrator AWS-Service , den Zugriff auf Ressourcen sicher zu AWS kontrollieren. IAM-Administratoren kontrollieren, wer authentifiziert (angemeldet) und autorisiert werden kann (über Berechtigungen verfügt), um Ressourcen zu verwenden. AWS Proton IAM ist ein Programm AWS-Service , das Sie ohne zusätzliche Kosten nutzen können.

Themen

Zielgruppe

Wie Sie AWS Identity and Access Management (IAM) verwenden, hängt von Ihrer Rolle ab:

- Servicebenutzer – Fordern Sie von Ihrem Administrator Berechtigungen an, wenn Sie nicht auf Features zugreifen können (siehe Problembehandlung bei AWS Proton Identität und Zugriff).
- Serviceadministrator – Bestimmen Sie den Benutzerzugriff und stellen Sie Berechtigungsanfragen (siehe Wie AWS Proton funktioniert mit IAM).
- IAM-Administrator – Schreiben Sie Richtlinien zur Zugriffsverwaltung (siehe Beispiele für identitätsbasierte Richtlinien für AWS Proton).

Authentifizierung mit Identitäten

Authentifizierung ist die Art und Weise, wie Sie sich AWS mit Ihren Identitätsdaten anmelden. Sie müssen sich als IAM-Benutzer authentifizieren oder eine IAM-Rolle annehmen. Root-Benutzer des AWS-Kontos

Sie können sich als föderierte Identität anmelden, indem Sie Anmeldeinformationen aus einer Identitätsquelle wie AWS IAM Identity Center (IAM Identity Center), Single Sign-On-Authentifizierung oder Anmeldeinformationen verwenden. Google/Facebook Weitere Informationen zum Anmelden finden Sie unter So melden Sie sich bei Ihrem AWS-Konto an im Benutzerhandbuch für AWS-Anmeldung .

AWS Bietet für den programmatischen Zugriff ein SDK und eine CLI zum kryptografischen Signieren von Anfragen. Weitere Informationen finden Sie unter AWS Signature Version 4 for API requests im IAM-Benutzerhandbuch.

AWS-Konto Root-Benutzer

Wenn Sie einen erstellen AWS-Konto, beginnen Sie mit einer Anmeldeidentität, dem sogenannten AWS-Konto Root-Benutzer, der vollständigen Zugriff auf alle AWS-Services Ressourcen hat. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Eine Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter Tasks that require root user credentials im IAM-Benutzerhandbuch.

Verbundidentität

Es hat sich bewährt, dass menschliche Benutzer für den Zugriff AWS-Services mithilfe temporärer Anmeldeinformationen einen Verbund mit einem Identitätsanbieter verwenden müssen.

Eine föderierte Identität ist ein Benutzer aus Ihrem Unternehmensverzeichnis, Ihrem Directory Service Web-Identitätsanbieter oder der AWS-Services mithilfe von Anmeldeinformationen aus einer Identitätsquelle zugreift. Verbundene Identitäten übernehmen Rollen, die temporäre Anmeldeinformationen bereitstellen.

Für die zentrale Zugriffsverwaltung empfehlen wir AWS IAM Identity Center. Weitere Informationen finden Sie unter Was ist IAM Identity Center? im AWS IAM Identity Center -Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein IAM-Benutzer ist eine Identität mit bestimmten Berechtigungen für eine einzelne Person oder Anwendung. Wir empfehlen die Verwendung temporärer Anmeldeinformationen anstelle von IAM-Benutzern mit langfristigen Anmeldeinformationen. Weitere Informationen finden Sie im IAM-Benutzerhandbuch unter Erfordern, dass menschliche Benutzer den Verbund mit einem Identitätsanbieter verwenden müssen, um AWS mithilfe temporärer Anmeldeinformationen darauf zugreifen zu können.

Eine IAM-Gruppe spezifiziert eine Sammlung von IAM-Benutzern und erleichtert die Verwaltung von Berechtigungen für große Gruppen von Benutzern. Weitere Informationen finden Sie unter Anwendungsfälle für IAM-Benutzer im IAM-Benutzerhandbuch.

IAM-Rollen

Eine IAM-Rolle ist eine Identität mit spezifischen Berechtigungen, die temporäre Anmeldeinformationen bereitstellt. Sie können eine Rolle übernehmen, indem Sie von einer Benutzer- zu einer IAM-Rolle (Konsole) wechseln AWS CLI oder einen AWS API-Vorgang aufrufen. Weitere Informationen finden Sie unter Methoden, um eine Rolle zu übernehmen im IAM-Benutzerhandbuch.

IAM-Rollen sind nützlich für den Verbundbenutzer-Zugriff, temporäre IAM-Benutzerberechtigungen, kontoübergreifenden Zugriff, serviceübergreifenden Zugriff und Anwendungen, die auf Amazon EC2 laufen. Weitere Informationen finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

Verwalten des Zugriffs mit Richtlinien

Sie kontrollieren den Zugriff, AWS indem Sie Richtlinien erstellen und diese an AWS Identitäten oder Ressourcen anhängen. Eine Richtlinie definiert Berechtigungen, wenn sie mit einer Identität oder Ressource verknüpft sind. AWS bewertet diese Richtlinien, wenn ein Principal eine Anfrage stellt. Die meisten Richtlinien werden AWS als JSON-Dokumente gespeichert. Weitere Informationen zu JSON-Richtliniendokumenten finden Sie unter Übersicht über JSON-Richtlinien im IAM-Benutzerhandbuch.

Mit Hilfe von Richtlinien legen Administratoren fest, wer Zugriff auf was hat, indem sie definieren, welches Prinzipal welche Aktionen auf welchen Ressourcenund unter welchen Bedingungendurchführen darf.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator erstellt IAM-Richtlinien und fügt sie zu Rollen hinzu, die die Benutzer dann übernehmen können. IAM-Richtlinien definieren Berechtigungen unabhängig von der Methode, die zur Ausführung der Operation verwendet wird.

Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität (Benutzer, Gruppe oder Rolle) anfügen können. Diese Richtlinien steuern, welche Aktionen Identitäten für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Definieren benutzerdefinierter IAM-Berechtigungen mit vom Kunden verwalteten Richtlinien im IAM-Benutzerhandbuch.

Identitätsbasierte Richtlinien können Inline-Richtlinien (direkt in eine einzelne Identität eingebettet) oder verwaltete Richtlinien (eigenständige Richtlinien, die mit mehreren Identitäten verbunden sind) sein. Informationen dazu, wie Sie zwischen verwalteten und Inline-Richtlinien wählen, finden Sie unter Choose between managed policies and inline policies im IAM-Benutzerhandbuch.

Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele hierfür sind Vertrauensrichtlinien für IAM-Rollen und Amazon S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben.

Ressourcenbasierte Richtlinien sind Richtlinien innerhalb dieses Diensts. Sie können AWS verwaltete Richtlinien von IAM nicht in einer ressourcenbasierten Richtlinie verwenden.

Weitere Richtlinientypen

AWS unterstützt zusätzliche Richtlinientypen, mit denen die maximalen Berechtigungen festgelegt werden können, die durch gängigere Richtlinientypen gewährt werden:

- Berechtigungsgrenzen – Eine Berechtigungsgrenze legt die maximalen Berechtigungen fest, die eine identitätsbasierte Richtlinie einer IAM-Entität erteilen kann. Weitere Informationen finden Sie unter Berechtigungsgrenzen für IAM-Entitäten im -IAM-Benutzerhandbuch.
- Richtlinien zur Dienstkontrolle (SCPs) — Geben Sie die maximalen Berechtigungen für eine Organisation oder Organisationseinheit in an AWS Organizations. Weitere Informationen finden Sie unter Service-Kontrollrichtlinien im AWS Organizations -Benutzerhandbuch.
- Richtlinien zur Ressourcenkontrolle (RCPs) — Legen Sie die maximal verfügbaren Berechtigungen für Ressourcen in Ihren Konten fest. Weitere Informationen finden Sie im AWS Organizations Benutzerhandbuch unter Richtlinien zur Ressourcenkontrolle (RCPs).
- Sitzungsrichtlinien – Sitzungsrichtlinien sind erweiterte Richtlinien, die als Parameter übergeben werden, wenn Sie eine temporäre Sitzung für eine Rolle oder einen Verbundbenutzer erstellen. Weitere Informationen finden Sie unter Sitzungsrichtlinien im IAM-Benutzerhandbuch.

Mehrere Richtlinientypen

Wenn für eine Anfrage mehrere Arten von Richtlinien gelten, sind die sich daraus ergebenden Berechtigungen schwieriger zu verstehen. Informationen darüber, wie AWS bestimmt wird, ob eine

Anfrage zulässig ist, wenn mehrere Richtlinientypen betroffen sind, finden Sie unter Bewertungslogik für Richtlinien im IAM-Benutzerhandbuch.

Wie AWS Proton funktioniert mit IAM

Bevor Sie IAM zur Verwaltung des Zugriffs auf verwenden, sollten Sie sich darüber informieren AWS Proton, mit welchen IAM-Funktionen Sie arbeiten können. AWS Proton

IAM-Funktionen, die Sie mit verwenden können AWS Proton

IAM-Feature	AWS Proton Unterstützung
Identitätsbasierte Richtlinien	Ja
Ressourcenbasierte Richtlinien	Nein
Richtlinienaktionen	Ja
Richtlinienressourcen	Ja
Bedingungsschlüssel für die Richtlinie	Ja
ACLs	Nein
ABAC (Tags in Richtlinien)	Ja
Temporäre Anmeldeinformationen	Ja
Prinzipalberechtigungen	Ja
Servicerollen	Ja
Service-verknüpfte Rollen	Ja

Einen allgemeinen Überblick darüber, wie die meisten IAM-Funktionen AWS-Services funktionieren AWS Proton und wie sie funktionieren AWS-Services , finden Sie im IAM-Benutzerhandbuch.

Identitätsbasierte Richtlinien für AWS Proton

Unterstützt Richtlinien auf Identitätsbasis: Ja

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Definieren benutzerdefinierter IAM-Berechtigungen mit vom Kunden verwalteten Richtlinien im IAM-Benutzerhandbuch.

Mit identitätsbasierten IAM-Richtlinien können Sie angeben, welche Aktionen und Ressourcen zugelassen oder abgelehnt werden. Darüber hinaus können Sie die Bedingungen festlegen, unter denen Aktionen zugelassen oder abgelehnt werden. Informationen zu sämtlichen Elementen, die Sie in einer JSON-Richtlinie verwenden, finden Sie in der IAM-Referenz für JSON-Richtlinienelemente im IAM-Benutzerhandbuch.

Beispiele für identitätsbasierte Richtlinien für AWS Proton

Beispiele für AWS Proton identitätsbasierte Richtlinien finden Sie unter. Beispiele für identitätsbasierte Richtlinien für AWS Proton

Ressourcenbasierte Richtlinien finden Sie in AWS Proton

Unterstützt ressourcenbasierte Richtlinien: Nein

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und Amazon-S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Zu den Prinzipalen können Konten, Benutzer, Rollen, Verbundbenutzer oder gehören. AWS-Services

Um kontoübergreifenden Zugriff zu ermöglichen, können Sie ein gesamtes Konto oder IAM-Entitäten in einem anderen Konto als Prinzipal in einer ressourcenbasierten Richtlinie angeben. Weitere Informationen finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

Richtlinienaktionen für AWS Proton

Unterstützt Richtlinienaktionen: Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das Element `Action` einer JSON-Richtlinie beschreibt die Aktionen, mit denen Sie den Zugriff in einer Richtlinie zulassen oder verweigern können. Nehmen Sie Aktionen in eine Richtlinie auf, um Berechtigungen zur Ausführung des zugehörigen Vorgangs zu erteilen.

Eine Liste der AWS Proton Aktionen finden Sie unter Aktionen definiert von AWS Proton in der Serviceautorisierungsreferenz.

Bei Richtlinienaktionen wird vor der Aktion das folgende Präfix AWS Proton verwendet:

```
proton
```

Um mehrere Aktionen in einer einzigen Anweisung anzugeben, trennen Sie sie mit Kommata:

```
"Action": [
      "proton:action1",
      "proton:action2"
         ]
```

Sie können auch Platzhalter verwenden, um mehrere Aktionen anzugeben. Beispielsweise können Sie alle Aktionen festlegen, die mit dem Wort `List` beginnen, einschließlich der folgenden Aktion:

```
"Action": "proton:List*"
```

Beispiele für AWS Proton identitätsbasierte Richtlinien finden Sie unter. Beispiele für identitätsbasierte Richtlinien für AWS Proton

Politische Ressourcen für AWS Proton

Unterstützt Richtlinienressourcen: Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das JSON-Richtlinienelement `Resource` gibt die Objekte an, auf welche die Aktion angewendet wird. Als Best Practice geben Sie eine Ressource mit dem zugehörigen Amazon-Ressourcennamen

(ARN) an. Verwenden Sie für Aktionen, die keine Berechtigungen auf Ressourcenebene unterstützen, einen Platzhalter (*), um anzugeben, dass die Anweisung für alle Ressourcen gilt.

```
"Resource": "*"
```

Eine Liste der AWS Proton Ressourcentypen und ihrer ARNs Eigenschaften finden Sie unter Resources defined by AWS Proton in der Service Authorization Reference. Informationen zu den Aktionen, mit denen Sie den ARN einzelner Ressourcen angeben können, finden Sie unter Von AWS Proton definierte Aktionen.

Beispiele für AWS Proton identitätsbasierte Richtlinien finden Sie unter. Beispiele für identitätsbasierte Richtlinien für AWS Proton

Bedingungsschlüssel für Richtlinien für AWS Proton

Unterstützt servicespezifische Richtlinienbedingungsschlüssel: Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer auf was Zugriff hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das Element `Condition` gibt an, wann Anweisungen auf der Grundlage definierter Kriterien ausgeführt werden. Sie können bedingte Ausdrücke erstellen, die Bedingungsoperatoren verwenden, z. B. ist gleich oder kleiner als, damit die Bedingung in der Richtlinie mit Werten in der Anforderung übereinstimmt. Eine Übersicht aller AWS globalen Bedingungsschlüssel finden Sie unter Kontextschlüssel für AWS globale Bedingungen im IAM-Benutzerhandbuch.

Eine Liste der AWS Proton Bedingungsschlüssel finden Sie unter Bedingungsschlüssel für AWS Proton in der Service Authorization Reference. Informationen zu den Aktionen und Ressourcen, mit denen Sie einen Bedingungsschlüssel verwenden können, finden Sie unter Aktionen definiert von AWS Proton.

Ein Beispiel für eine condition-key-based Richtlinie zur Beschränkung des Zugriffs auf eine Ressource finden Sie unterBeispiele für Richtlinien auf der Grundlage von Bedingungsschlüsseln für AWS Proton.

Zugriffskontrolllisten (ACLs) in AWS Proton

Unterstützt ACLs: Nein

Zugriffskontrolllisten (ACLs) steuern, welche Principals (Kontomitglieder, Benutzer oder Rollen) über Zugriffsberechtigungen für eine Ressource verfügen. ACLs ähneln ressourcenbasierten Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

Zugriffskontrolllisten (ACLs) sind Listen von Empfängern, die Sie Ressourcen zuordnen können. Sie erteilen Konten Berechtigungen für den Zugriff auf die Ressource, auf die sie sich beziehen.

Attributbasierte Zugriffskontrolle (ABAC) mit AWS Proton

Unterstützt ABAC (Tags in Richtlinien): Ja

Die attributbasierte Zugriffskontrolle (ABAC) ist eine Autorisierungsstrategie, bei der Berechtigungen basierend auf Attributen, auch als Tags bezeichnet, definiert werden. Sie können Tags an IAM-Entitäten und AWS -Ressourcen anhängen und dann ABAC-Richtlinien entwerfen, die Operationen zulassen, wenn das Tag des Prinzipals mit dem Tag auf der Ressource übereinstimmt.

Um den Zugriff auf der Grundlage von Tags zu steuern, geben Sie im Bedingungselement einer Richtlinie Tag-Informationen an, indem Sie die Schlüssel `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`*, oder Bedingung `aws:TagKeys` verwenden.

Wenn ein Service alle drei Bedingungsschlüssel für jeden Ressourcentyp unterstützt, lautet der Wert für den Service Ja. Wenn ein Service alle drei Bedingungsschlüssel für nur einige Ressourcentypen unterstützt, lautet der Wert Teilweise.

Weitere Informationen zu ABAC finden Sie unter Definieren von Berechtigungen mit ABAC-Autorisierung im IAM-Benutzerhandbuch. Um ein Tutorial mit Schritten zur Einstellung von ABAC anzuzeigen, siehe Attributbasierte Zugriffskontrolle (ABAC) verwenden im IAM-Benutzerhandbuch.

Weitere Informationen zum Markieren von AWS Proton Ressourcen finden Sie unter. AWS Proton Ressourcen und Tagging

Temporäre Anmeldeinformationen verwenden mit AWS Proton

Unterstützt temporäre Anmeldeinformationen: Ja

Temporäre Anmeldeinformationen ermöglichen kurzfristigen Zugriff auf AWS Ressourcen und werden automatisch erstellt, wenn Sie einen Verbund verwenden oder die Rollen wechseln. AWS empfiehlt, temporäre Anmeldeinformationen dynamisch zu generieren, anstatt langfristige Zugriffsschlüssel zu verwenden. Weitere Informationen finden Sie unter Temporäre Anmeldeinformationen in IAM und AWS-Services , die mit IAM funktionieren im IAM-Benutzerhandbuch.

Serviceübergreifende Prinzipalberechtigungen für AWS Proton

Unterstützt Forward Access Sessions (FAS): Ja

Forward Access Sessions (FAS) verwenden die Berechtigungen des Prinzipals, der einen aufruft AWS-Service, in Kombination mit der Anforderung, Anfragen an nachgelagerte Dienste AWS-Service zu stellen. Einzelheiten zu den Richtlinien für FAS-Anforderungen finden Sie unter Zugriffssitzungen weiterleiten.

Servicerollen für AWS Proton

Unterstützt Servicerollen: Ja

Eine Servicerolle ist eine IAM-Rolle, die ein Service annimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.

Weitere Informationen finden Sie unter AWS Proton Beispiele für Richtlinien für IAM-Servicerollen.

> **Warning**
>
> Das Ändern der Berechtigungen für eine Servicerolle kann zu AWS Proton Funktionseinschränkungen führen. Bearbeiten Sie Servicerollen nur, AWS Proton wenn Sie dazu eine Anleitung erhalten.

Dienstbezogene Rollen für AWS Proton

Unterstützt serviceverknüpfte Rollen: Ja

Eine dienstbezogene Rolle ist eine Art von Servicerolle, die mit einer verknüpft ist. AWS-Service Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Dienstbezogene Rollen werden in Ihrem Dienst angezeigt AWS-Konto und gehören dem Dienst. Ein IAM-Administrator kann die Berechtigungen für Service-verknüpfte Rollen anzeigen, aber nicht bearbeiten.

Weitere Informationen finden Sie unter Verwenden von serviceverknüpften Rollen für AWS Proton.

Richtlinienbeispiele für AWS Proton

In den folgenden Abschnitten finden Sie Beispiele für AWS Proton IAM-Richtlinien.

Topics

Beispiele für identitätsbasierte Richtlinien für AWS Proton

Benutzer und Rollen haben standardmäßig nicht die Berechtigung, AWS Proton -Ressourcen zu erstellen oder zu ändern. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen.

Informationen dazu, wie Sie unter Verwendung dieser beispielhaften JSON-Richtliniendokumente eine identitätsbasierte IAM-Richtlinie erstellen, finden Sie unter Erstellen von IAM-Richtlinien (Konsole) im IAM-Benutzerhandbuch.

Einzelheiten zu Aktionen und Ressourcentypen, die von definiert wurden AWS Proton, einschließlich des Formats von ARNs für die einzelnen Ressourcentypen, finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für AWS Proton in der Referenz zur Serviceautorisierung.

Themen

Best Practices für Richtlinien

Identitätsbasierte Richtlinien legen fest, ob jemand AWS Proton Ressourcen in Ihrem Konto erstellen, darauf zugreifen oder sie löschen kann. Dies kann zusätzliche Kosten für Ihr verursachen AWS-Konto. Beachten Sie beim Erstellen oder Bearbeiten identitätsbasierter Richtlinien die folgenden Richtlinien und Empfehlungen:

- Erste Schritte mit AWS verwalteten Richtlinien und Umstellung auf Berechtigungen mit den geringsten Rechten — Verwenden Sie die AWS verwalteten Richtlinien, die Berechtigungen für viele gängige Anwendungsfälle gewähren, um damit zu beginnen, Ihren Benutzern und Workloads Berechtigungen zu gewähren. Sie sind in Ihrem verfügbar. AWS-Konto Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie vom AWS Kunden verwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind. Weitere Informationen

finden Sie unter Von AWS verwaltete Richtlinien oder Von AWS verwaltete Richtlinien für Auftragsfunktionen im IAM-Benutzerhandbuch.

- Anwendung von Berechtigungen mit den geringsten Rechten – Wenn Sie mit IAM-Richtlinien Berechtigungen festlegen, gewähren Sie nur die Berechtigungen, die für die Durchführung einer Aufgabe erforderlich sind. Sie tun dies, indem Sie die Aktionen definieren, die für bestimmte Ressourcen unter bestimmten Bedingungen durchgeführt werden können, auch bekannt als die geringsten Berechtigungen. Weitere Informationen zur Verwendung von IAM zum Anwenden von Berechtigungen finden Sie unter Richtlinien und Berechtigungen in IAM im IAM-Benutzerhandbuch.
- Verwenden von Bedingungen in IAM-Richtlinien zur weiteren Einschränkung des Zugriffs – Sie können Ihren Richtlinien eine Bedingung hinzufügen, um den Zugriff auf Aktionen und Ressourcen zu beschränken. Sie können beispielsweise eine Richtlinienbedingung schreiben, um festzulegen, dass alle Anforderungen mithilfe von SSL gesendet werden müssen. Sie können auch Bedingungen verwenden, um Zugriff auf Serviceaktionen zu gewähren, wenn diese für einen bestimmten Zweck verwendet werden AWS-Service, z. CloudFormation B. Weitere Informationen finden Sie unter IAM-JSON-Richtlinienelemente: Bedingung im IAM-Benutzerhandbuch.
- Verwenden von IAM Access Analyzer zur Validierung Ihrer IAM-Richtlinien, um sichere und funktionale Berechtigungen zu gewährleisten – IAM Access Analyzer validiert neue und vorhandene Richtlinien, damit die Richtlinien der IAM-Richtliniensprache (JSON) und den bewährten IAM-Methoden entsprechen. IAM Access Analyzer stellt mehr als 100 Richtlinienprüfungen und umsetzbare Empfehlungen zur Verfügung, damit Sie sichere und funktionale Richtlinien erstellen können. Weitere Informationen finden Sie unter Richtlinienvalidierung mit IAM Access Analyzer im IAM-Benutzerhandbuch.
- Multi-Faktor-Authentifizierung (MFA) erforderlich — Wenn Sie ein Szenario haben, das IAM-Benutzer oder einen Root-Benutzer in Ihrem System erfordert AWS-Konto, aktivieren Sie MFA für zusätzliche Sicherheit. Um MFA beim Aufrufen von API-Vorgängen anzufordern, fügen Sie Ihren Richtlinien MFA-Bedingungen hinzu. Weitere Informationen finden Sie unter Sicherer API-Zugriff mit MFA im IAM-Benutzerhandbuch.

Weitere Informationen zu bewährten Methoden in IAM finden Sie unter Best Practices für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Links zu Beispielen für identitätsbasierte Richtlinien für AWS Proton

Links zu beispielhaften Beispielen für identitätsbasierte Richtlinien für AWS Proton

- AWS verwaltete Richtlinien für AWS Proton

- AWS Proton Beispiele für Richtlinien für IAM-Servicerollen
- Beispiele für Richtlinien auf der Grundlage von Bedingungsschlüsseln für AWS Proton

AWS Proton Beispiele für Richtlinien für IAM-Servicerollen

Administratoren besitzen und verwalten die Ressourcen, die AWS Proton erstellt werden, wie in der Umgebung und den Dienstvorlagen definiert. Sie ordnen ihrem Konto IAM-Servicerollen zu, die es ihnen ermöglichen, Ressourcen in ihrem Namen AWS Proton zu erstellen. Administratoren stellen die IAM-Rollen und AWS Key Management Service Schlüssel für Ressourcen bereit, die später den Entwicklern gehören und von diesen verwaltet werden, wenn AWS Proton sie ihre Anwendung als AWS Proton Dienst in einer Umgebung bereitstellen. AWS Proton Weitere Informationen AWS KMS zur Datenverschlüsselung finden Sie unter. Datenschutz in AWS Proton

Eine Servicerolle ist eine Amazon Web Services (IAM) -Rolle, mit der AWS Proton Sie Ressourcen in Ihrem Namen aufrufen können. Wenn Sie eine Service-Rolle festlegen, verwendet AWS Proton die Anmeldeinformationen der Rolle. Verwenden Sie eine Servicerolle, um explizit anzugeben, welche Aktionen ausgeführt AWS Proton werden können.

Sie erstellen die Service-Rolle und die Berechtigungsrichtlinie mit dem IAM-Service. Weitere Informationen zum Erstellen einer Servicerolle finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen für einen AWS Dienst im IAM-Benutzerhandbuch.

AWS Proton Servicerolle für die Bereitstellung mit CloudFormation

Als Mitglied des Plattformteams können Sie als Administrator eine AWS Proton Servicerolle erstellen und sie bei der Erstellung einer Umgebung als CloudFormation Servicerolle der Umgebung bereitstellen (der `protonServiceRoleArn` Parameter der CreateEnvironmentAPI-Aktion). AWS Proton Mit dieser Rolle können AWS Proton Sie in Ihrem Namen API-Aufrufe an andere Dienste tätigen, wenn die Umgebung oder eine der darin ausgeführten Dienstinstanzen AWS verwaltetes Provisioning verwenden und AWS CloudFormation die Infrastruktur bereitstellen.

Wir empfehlen Ihnen, die folgende IAM-Rollen- und Vertrauensrichtlinie für Ihre AWS Proton Servicerolle zu verwenden. Wenn Sie die AWS Proton Konsole verwenden, um eine Umgebung zu erstellen, und sich dafür entscheiden, eine neue Rolle zu erstellen, ist dies die Richtlinie, die zu der für Sie erstellten Servicerolle AWS Proton hinzugefügt wird. Wenn Sie den Umfang der Berechtigungen für diese Richtlinie einschränken, sollten Sie bedenken, dass sie bei Fehlern AWS Proton fehlschlägt. `Access Denied`

> **Important**
>
> Beachten Sie, dass die in den folgenden Beispielen gezeigten Richtlinien jedem, der eine Vorlage für Ihr Konto registrieren kann, Administratorrechte gewähren. Da wir nicht wissen, welche Ressourcen Sie in Ihren AWS Proton Vorlagen definieren werden, verfügen diese Richtlinien über umfassende Zugriffsrechte. Wir empfehlen, dass Sie die Berechtigungen auf die spezifischen Ressourcen beschränken, die in Ihren Umgebungen bereitgestellt werden.

AWS Proton Beispiel für eine Servicerollenrichtlinie für CloudFormation

Ersetze es *123456789012* durch deine AWS-Konto ID.

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 Vertrauensrichtlinie für Dienste

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/*"
      }
    }
  }
}
```

Rollenrichtlinie für AWS verwaltete Bereitstellungsdienste mit eingeschränktem Geltungsbereich

Im Folgenden finden Sie ein Beispiel für eine AWS Proton Servicerollenrichtlinie mit eingeschränktem Geltungsbereich, die Sie verwenden können, wenn Sie Dienste nur für die Bereitstellung von S3-Ressourcen benötigen AWS Proton .

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 Servicerolle für die Bereitstellung CodeBuild

Als Mitglied des Plattformteams können Sie als Administrator eine AWS Proton Servicerolle erstellen und sie bei der Erstellung einer Umgebung als CodeBuild Servicerolle der Umgebung bereitstellen (der `codebuildRoleArn` Parameter der CreateEnvironmentAPI-Aktion). AWS Proton Diese Rolle ermöglicht es AWS Proton , API-Aufrufe an andere Dienste in Ihrem Namen zu tätigen, wenn die Umgebung oder eine der darin ausgeführten Dienstinstanzen CodeBuild Provisioning zur Bereitstellung der Infrastruktur verwenden.

Wenn Sie die AWS Proton Konsole verwenden, um eine Umgebung zu erstellen, und sich dafür entscheiden, eine neue Rolle zu erstellen, AWS Proton fügt der für Sie erstellten Servicerolle eine Richtlinie mit Administratorrechten hinzu. Wenn Sie Ihre eigene Rolle erstellen und den Umfang der Berechtigungen einschränken, denken Sie daran, dass AWS Proton dies bei `Access Denied` Fehlern fehlschlägt.

> **Important**
>
> Beachten Sie, dass die Richtlinien, die AWS Proton an die Rollen angehängt werden, die das Programm für Sie erstellt, jedem, der eine Vorlage für Ihr Konto registrieren kann, Administratorrechte gewähren. Da wir nicht wissen, welche Ressourcen Sie in Ihren AWS Proton Vorlagen definieren werden, verfügen diese Richtlinien über umfassende Zugriffsrechte. Wir empfehlen, dass Sie die Berechtigungen auf die spezifischen Ressourcen beschränken, die in Ihren Umgebungen bereitgestellt werden.

AWS Proton Beispiel für eine Servicerollenrichtlinie für CodeBuild

Das folgende Beispiel bietet Berechtigungen für CodeBuild die Bereitstellung von Ressourcen mithilfe von AWS Cloud Development Kit (AWS CDK).

Ersetze es *123456789012* durch deine AWS-Konto ID.

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 Vertrauensrichtlinie

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 Rollen im Pipeline-Dienst

Für die Bereitstellung von Service-Pipelines sind AWS Proton Berechtigungen erforderlich, um API-Aufrufe an andere Dienste zu tätigen. Die erforderlichen Servicerollen ähneln den Servicerollen, die Sie beim Erstellen von Umgebungen angeben. Die Rollen zum Erstellen von Pipelines werden

jedoch von allen Diensten in Ihrem AWS Konto gemeinsam genutzt, und Sie geben diese Rollen als Kontoeinstellungen in der Konsole oder über die UpdateAccountSettingsAPI-Aktion an.

Wenn Sie die AWS Proton Konsole verwenden, um die Kontoeinstellungen zu aktualisieren und sich dafür entscheiden, eine neue Rolle für die CloudFormation oder die CodeBuild Dienstrollen zu erstellen, entsprechen die Richtlinien, die zu den für Sie erstellten Servicerollen AWS Proton hinzugefügt werden, den Richtlinien, die in den vorherigen Abschnitten beschrieben wurden, AWS-verwaltete Bereitstellungsrolle undCodeBuild Rolle bei der Bereitstellung. Denken Sie bei der Einschränkung des Berechtigungsbereichs für diese Richtlinie daran, dass diese bei Fehlern AWS Proton fehlschlägt. `Access Denied`

> **Important**
>
> Beachten Sie, dass die Beispielrichtlinien in den vorherigen Abschnitten jedem, der eine Vorlage für Ihr Konto registrieren kann, Administratorrechte gewähren. Da wir nicht wissen, welche Ressourcen Sie in Ihren AWS Proton Vorlagen definieren werden, verfügen diese Richtlinien über umfassende Zugriffsrechte. Wir empfehlen, dass Sie die Berechtigungen auf die spezifischen Ressourcen beschränken, die in Ihren Pipelines bereitgestellt werden.

AWS Proton Rolle der Komponente

Als Mitglied des Plattformteams können Sie als Administrator eine AWS Proton Servicerolle erstellen und sie bei der Erstellung einer Umgebung als CloudFormation Komponentenrolle der Umgebung bereitstellen (der `componentRoleArn` Parameter der CreateEnvironmentAPI-Aktion). AWS Proton Diese Rolle umfasst die Infrastruktur, die direkt definierte Komponenten bereitstellen können. Weitere Informationen zu Komponenten finden Sie unterKomponenten.

Die folgende Beispielrichtlinie unterstützt die Erstellung einer direkt definierten Komponente, die einen Amazon Simple Storage Service (Amazon S3) -Bucket und eine zugehörige Zugriffsrichtlinie bereitstellt.

AWS Proton Beispiel für eine Rollenrichtlinie für eine Komponente

Ersetze es *123456789012* durch deine AWS-Konto ID.

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 Vertrauensrichtlinie für Komponenten

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/*"
      }
    }
  }
}
```

Beispiele für Richtlinien auf der Grundlage von Bedingungsschlüsseln für AWS Proton

Das folgende Beispiel für eine IAM-Richtlinie verweigert den Zugriff auf AWS Proton Aktionen, die den im Block angegebenen Vorlagen entsprechen. `Condition` Beachten Sie, dass diese Bedingungsschlüssel nur von den Aktionen unterstützt werden, die unter Aktionen, Ressourcen und Bedingungsschlüssel für aufgeführt sind. AWS Proton Um Berechtigungen für andere Aktionen zu verwalten, müssen Sie z. `DeleteEnvironmentTemplate` B. die Zugriffskontrolle auf Ressourcenebene verwenden.

Beispielrichtlinie, die AWS Proton Vorlagenaktionen für eine bestimmte Vorlage verweigert:

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"]
                }
            }
        }
    ]
}
```

In der nächsten Beispielrichtlinie verweigert die erste Anweisung auf Ressourcenebene den Zugriff auf AWS Proton Vorlagenaktionen`ListServiceTemplates`, die nicht mit der im Block aufgeführten Dienstvorlage übereinstimmen. `Resource` Die zweite Anweisung verweigert den Zugriff auf AWS Proton Aktionen, die der im Block aufgeführten Vorlage entsprechen. `Condition`

Beispielrichtlinie, die AWS Proton Aktionen verweigert, die einer bestimmten Vorlage entsprechen:

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"
                    ]
                }
            }
        }
    ]
}
```

Das letzte Richtlinienbeispiel erlaubt AWS Proton Entwickleraktionen, die der spezifischen Dienstvorlage entsprechen, die im `Condition` Block aufgeführt ist.

Beispielrichtlinie, um AWS Proton Entwickleraktionen zuzulassen, die einer bestimmten Vorlage entsprechen:

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 verwaltete Richtlinien für AWS Proton

Um Benutzern, Gruppen und Rollen Berechtigungen hinzuzufügen, ist es einfacher, AWS verwaltete Richtlinien zu verwenden, als Richtlinien selbst zu schreiben. Es erfordert Zeit und Fachwissen, um von Kunden verwaltete IAM-Richtlinien zu erstellen, die Ihrem Team nur die benötigten Berechtigungen bieten. Um schnell loszulegen, können Sie unsere AWS verwalteten Richtlinien verwenden. Diese Richtlinien decken allgemeine Anwendungsfälle ab und sind in Ihrem AWS-Konto verfügbar. Weitere Informationen zu AWS verwalteten Richtlinien finden Sie im IAM-Benutzerhandbuch unter AWS Verwaltete Richtlinien.

AWS-Services verwalten und aktualisieren Sie AWS verwaltete Richtlinien. Sie können die Berechtigungen in AWS verwalteten Richtlinien nicht ändern. Services fügen einer von AWS verwalteten Richtlinien gelegentlich zusätzliche Berechtigungen hinzu, um neue Features zu unterstützen. Diese Art von Update betrifft alle Identitäten (Benutzer, Gruppen und Rollen), an welche die Richtlinie angehängt ist. Services aktualisieren eine von AWS verwaltete Richtlinie am ehesten, ein neues Feature gestartet wird oder neue Vorgänge verfügbar werden. Dienste entfernen keine Berechtigungen aus einer AWS verwalteten Richtlinie, sodass durch Richtlinienaktualisierungen Ihre bestehenden Berechtigungen nicht beeinträchtigt werden.

AWS Unterstützt außerdem verwaltete Richtlinien für Jobfunktionen, die sich über mehrere Dienste erstrecken. Die ReadOnlyAccess AWS verwaltete Richtlinie bietet beispielsweise nur Lesezugriff auf alle Ressourcen AWS-Services . Wenn ein Service ein neues Feature startet, fügt AWS schreibgeschützte Berechtigungen für neue Vorgänge und Ressourcen hinzu. Eine Liste und Beschreibungen der Richtlinien für Auftragsfunktionen finden Sie in Verwaltete AWS -Richtlinien für Auftragsfunktionen im IAM-Leitfaden.

AWS Proton bietet verwaltete IAM-Richtlinien und Vertrauensstellungen, die Sie Benutzern, Gruppen oder Rollen zuordnen können, sodass Sie Ressourcen und API-Operationen unterschiedlich steuern können. Sie können diese Richtlinien direkt anwenden oder als Ausgangspunkt nutzen, um eigene Richtlinien zu erstellen.

Die folgende Vertrauensstellung wird für jede der AWS Proton verwalteten Richtlinien verwendet.

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 verwaltete Richtlinie: AWSProton FullAccess

Sie können eine Verbindung `AWSProtonFullAccess` zu Ihren IAM-Entitäten herstellen. AWS Proton ordnet diese Richtlinie auch einer Servicerolle zu, mit der Sie Aktionen AWS Proton in Ihrem Namen ausführen können.

Diese Richtlinie gewährt Administratorberechtigungen, die vollen Zugriff auf AWS Proton Aktionen und eingeschränkten Zugriff auf andere AWS Dienstaktionen ermöglichen, AWS Proton abhängig von.

Die Richtlinie umfasst die folgenden Namespaces für wichtige Aktionen:

- `proton`— Ermöglicht Administratoren vollen Zugriff auf. AWS Proton APIs
- `iam`— Ermöglicht Administratoren die Übergabe von Rollen an AWS Proton. Dies ist erforderlich, AWS Proton damit im Namen des Administrators API-Aufrufe an andere Dienste getätigt werden können.

- `kms`— Ermöglicht Administratoren, einem vom Kunden verwalteten Schlüssel einen Zuschuss hinzuzufügen.
- `codeconnections`— Ermöglicht Administratoren, Codeverbindungen aufzulisten und weiterzugeben, sodass sie von AWS Proton verwendet werden können.

Weitere Informationen finden Sie unter AWSProtonFullAccess.

AWS verwaltete Richtlinie: AWSProton DeveloperAccess

Sie können eine Verbindung `AWSProtonDeveloperAccess` zu Ihren IAM-Entitäten herstellen. AWS Proton ordnet diese Richtlinie auch einer Servicerolle zu, mit der Sie Aktionen AWS Proton in Ihrem Namen ausführen können.

Diese Richtlinie gewährt Berechtigungen, die eingeschränkten Zugriff auf AWS Proton Aktionen und andere AWS Aktionen ermöglichen, die AWS Proton davon abhängen. Der Umfang dieser Berechtigungen ist darauf ausgelegt, die Rolle eines Entwicklers zu unterstützen, der AWS Proton Dienste entwickelt und bereitstellt.

Diese Richtlinie gewährt keinen Zugriff auf die Erstellung, Löschung und Aktualisierung APIs von AWS Proton Vorlagen und Umgebungen. Wenn Entwickler noch eingeschränktere Berechtigungen benötigen, als diese Richtlinie vorsieht, empfehlen wir, eine benutzerdefinierte Richtlinie zu erstellen, die so begrenzt ist, dass sie nur die geringsten Rechte gewährt.

Die Richtlinie umfasst die folgenden Namespaces für wichtige Aktionen:

- `proton`— Ermöglicht Mitwirkenden den Zugriff auf eine begrenzte Anzahl von. AWS Proton APIs
- `codeconnections`— Erlaubt Mitwirkenden, Codeverbindungen aufzulisten und weiterzugeben, damit sie von AWS Proton verwendet werden können.

Weitere Informationen finden Sie unter AWSProtonDeveloperAccess.

AWS verwaltete Richtlinie: AWSProton ReadOnlyAccess

Sie können eine Verbindung `AWSProtonReadOnlyAccess` zu Ihren IAM-Entitäten herstellen. AWS Proton ordnet diese Richtlinie auch einer Servicerolle zu, mit der Sie Aktionen AWS Proton in Ihrem Namen ausführen können.

Diese Richtlinie gewährt Berechtigungen, die nur Lesezugriff auf AWS Proton Aktionen und eingeschränkten Lesezugriff auf andere AWS Dienstaktionen ermöglichen, abhängig von. AWS Proton

Die Richtlinie umfasst die folgenden Namespaces für wichtige Aktionen:

- `proton`— Erlaubt Mitwirkenden nur Lesezugriff auf. AWS Proton APIs

Weitere Informationen finden Sie unter AWSProtonReadOnlyAccess.

AWS verwaltete Richtlinie: AWSProton SyncServiceRolePolicy

AWS Proton fügt diese Richtlinie der AWSServiceRoleForProtonSyncdienstbezogenen Rolle hinzu, die die AWS Proton Vorlagensynchronisierung ermöglicht.

Diese Richtlinie gewährt Berechtigungen, die eingeschränkten Zugriff auf AWS Proton Aktionen und andere AWS Dienstaktionen ermöglichen, die davon AWS Proton abhängen.

Die Richtlinie umfasst die folgenden Namespaces für wichtige Aktionen:

- `proton`— Ermöglicht den eingeschränkten AWS Proton Synchronisierungszugriff auf. AWS Proton APIs
- `codeconnections`— Ermöglicht die AWS Proton Synchronisierung mit eingeschränktem Zugriff auf CodeConnections APIs.

Weitere Informationen finden Sie unter AWSProtonSyncServiceRolePolicy.

AWS verwaltete Richtlinie: AWSProton CodeBuildProvisioningBasicAccess

Permissions CodeBuild muss einen Build für die AWS Proton CodeBuild Bereitstellung ausführen. Sie können es `AWSProtonCodeBuildProvisioningBasicAccess` an Ihre CodeBuild Provisioning-Rolle anhängen.

Diese Richtlinie gewährt die Mindestberechtigungen, damit die AWS Proton CodeBuild Bereitstellung funktioniert. Sie gewährt Berechtigungen, die das CodeBuild Generieren von Build-Logs ermöglichen. Es gewährt Proton auch die Erlaubnis, Benutzern Infrastructure as Code (IaC) -Ausgaben zur Verfügung zu AWS Proton stellen. Es bietet keine Berechtigungen, die von IaC-Tools zur Verwaltung der Infrastruktur benötigt werden.

Die Richtlinie umfasst die folgenden Namespaces für wichtige Aktionen:

- `logs`- Ermöglicht das Generieren von CodeBuild Build-Logs. Ohne diese Erlaubnis kann der Start nicht ausgeführt CodeBuild werden.
- `proton`- Ermöglicht es einem CodeBuild Provisioning-Befehl, die Aktualisierung der IaaC-Ausgaben für eine bestimmte AWS Proton Ressource aufzurufen`aws proton notify-resource-deployment-status-change`.

Weitere Informationen finden Sie unter AWSProtonCodeBuildProvisioningBasicAccess.

AWS verwaltete Richtlinie: AWSProton CodeBuildProvisioningServiceRolePolicy

AWS Proton ordnet diese Richtlinie der AWSServiceRoleForProtonCodeBuildProvisioningdienstbezogenen Rolle zu, die eine CodeBuild basierte AWS Proton Bereitstellung ermöglicht.

Diese Richtlinie gewährt Berechtigungen, die einen eingeschränkten Zugriff auf AWS Dienstaktionen ermöglichen, AWS Proton abhängig von.

Die Richtlinie umfasst die folgenden Namespaces für wichtige Aktionen:

- `cloudformation`— Ermöglicht die AWS Proton CodeBuild basierte Bereitstellung mit eingeschränktem Zugriff auf. CloudFormation APIs
- `codebuild`— Ermöglicht AWS Proton CodeBuild eingeschränktes Provisioning mit eingeschränktem Zugriff auf. CodeBuild APIs
- `iam`— Ermöglicht Administratoren die Übergabe von Rollen an AWS Proton. Dies ist erforderlich, AWS Proton damit im Namen des Administrators API-Aufrufe an andere Dienste getätigt werden können.
- `servicequotas`— Ermöglicht AWS Proton die Überprüfung des Limits für CodeBuild gleichzeitige Builds, wodurch ein ordnungsgemäßes Build-Queuing gewährleistet wird.

Weitere Informationen finden Sie unter AWSProtonCodeBuildProvisioningServiceRolePolicy.

AWS verwaltete Richtlinie: AWSProton ServiceGitSyncServiceRolePolicy

AWS Proton ordnet diese Richtlinie der AWSServiceRoleForProtonServiceSyncdienstbezogenen Rolle zu, mit der die Dienstsynchronisierung durchgeführt AWS Proton werden kann.

Diese Richtlinie gewährt Berechtigungen, die eingeschränkten Zugriff auf AWS Proton Aktionen und andere AWS Dienstaktionen ermöglichen, die davon AWS Proton abhängen.

Die Richtlinie umfasst die folgenden Namespaces für wichtige Aktionen:

- `proton`— Ermöglicht den eingeschränkten AWS Proton Synchronisierungszugriff auf. AWS Proton APIs

Weitere Informationen finden Sie unter AWSProtonServiceGitSyncServiceRolePolicy.

AWS Proton Aktualisierungen der AWS verwalteten Richtlinien

Hier finden Sie Informationen zu Aktualisierungen AWS verwalteter Richtlinien, die AWS Proton seit Beginn der Nachverfolgung dieser Änderungen durch diesen Dienst vorgenommen wurden. Abonnieren Sie den RSS-Feed auf der Seite AWS Proton Dokumentenverlauf, um automatische Benachrichtigungen über Änderungen an dieser Seite zu erhalten.

Änderungen	Beschreibung	Datum
AWSProtonCodeBuildProvisioningServiceRolePolicy – Aktualisierung auf eine bestehende Richtlinie	Die verwaltete Richtlinie für die dienstbezogene Rolle, die die Durchführung einer CodeBuild basierten Bereitstellung ermöglicht AWS Proton , gewährt jetzt Berechtigungen zum Aufrufen von Aktionen CloudFormation `TagResource` und `UntagResource` API-Aktionen. Diese Berechtigungen sind erforderlich, um Tagging-Vorgänge für Ressourcen durchzuführen.	15. Juni 2024
AWSProtonFullAccess – Aktualisierung auf eine bestehende Richtlinie	Die verwaltete Richtlinie für die dienstverknüpfte Rolle zur Verwendung von Git-Synchronisierung mit Git-Repositorys wurde für	25. April 2024

Änderungen	Beschreibung	Datum
	Ressourcen mit beiden Dienstpräfixen aktualisiert. Weitere Informationen finden Sie unter Verwenden von serviceverknüpften Rollen für AWS CodeConnections und verwaltete Richtlinien.	
AWSProtonDeveloperAccess – Aktualisierung auf eine bestehende Richtlinie	Die verwaltete Richtlinie für die dienstverknüpfte Rolle zur Verwendung von Git-Synchronisierung mit Git-Repositorys wurde für Ressourcen mit beiden Dienstpräfixen aktualisiert. Weitere Informationen finden Sie unter Verwenden von serviceverknüpften Rollen für AWS CodeConnections und verwaltete Richtlinien.	25. April 2024
AWSProtonSyncServiceRolePolicy – Aktualisierung auf eine bestehende Richtlinie	Die verwaltete Richtlinie für die dienstverknüpfte Rolle zur Verwendung von Git-Synchronisierung mit Git-Repositorys wurde für Ressourcen mit beiden Dienstpräfixen aktualisiert. Weitere Informationen finden Sie unter Verwenden von serviceverknüpften Rollen für AWS CodeConnections und verwaltete Richtlinien.	25. April 2024

Änderungen	Beschreibung	Datum
AWSProtonCodeBuildProvisioningServiceRolePolicy – Aktualisierung auf eine bestehende Richtlinie	AWS Proton Diese Richtlinie wurde aktualisiert, um Berechtigungen hinzuzufügen, um sicherzustellen, dass Konten über das erforderliche Limit für CodeBuild gleichzeitige Builds verfügen, um Provisioning verwenden zu können CodeBuild .	12. Mai 2023
AWSProtonServiceGitSyncServiceRolePolicy – Neue Richtlinie	AWS Proton hat eine neue Richtlinie hinzugefügt, die die Durchführung der AWS Proton Dienstsynchronisierung ermöglicht. Die Richtlinie wird in der AWSServiceRoleForProtonServiceSyncdienstbezogenen Rolle verwendet.	31. März 2023
AWSProtonDeveloperAccess – Aktualisierung auf eine bestehende Richtlinie	AWS Proton hat eine neue `GetResourcesSummary` Aktion hinzugefügt, mit der Sie eine Zusammenfassung Ihrer Vorlagen, bereitgestellten Vorlagenressourcen und veralteten Ressourcen anzeigen können.	18. November 2022

Änderungen	Beschreibung	Datum
AWSProtonReadOnlyAccess – Aktualisierung auf eine bestehende Richtlinie	AWS Proton hat eine neue `GetResourcesSummary` Aktion hinzugefügt, mit der Sie eine Zusammenfassung Ihrer Vorlagen, bereitgestellten Vorlagenressourcen und veralteten Ressourcen anzeigen können.	18. November 2022
AWSProtonCodeBuildProvisioningBasicAccess – Neue Richtlinie	AWS Proton hat eine neue Richtlinie hinzugefügt, CodeBuild die die zum Ausführen eines Builds für die AWS Proton CodeBuild Bereitstellung erforderlichen Berechtigungen gewährt.	16. November 2022
AWSProtonSyncServiceRolePolicy – Neue Richtlinie	AWS Proton hat eine neue Richtlinie hinzugefügt, die die Durchführung von Vorgängen im Zusammenhang mit der CodeBuild basierten Bereitstellung ermöglicht AWS Proton . Die Richtlinie wird in der AWSServiceRoleForProtonCodeBuildProvisioningserviceverknüpften Rolle verwendet.	02. September 2022

Änderungen	Beschreibung	Datum
AWSProtonFullAccess – Aktualisierung auf eine bestehende Richtlinie	AWS Proton hat diese Richtlinie aktualisiert, um Zugriff auf neue AWS Proton API-Operationen zu gewähren und Berechtigungsprobleme für einige AWS Proton Konsolenoperationen zu beheben.	30. März 2022
AWSProtonDeveloperAccess – Aktualisierung auf eine bestehende Richtlinie	AWS Proton Diese Richtlinie wurde aktualisiert, um Zugriff auf neue AWS Proton API-Operationen zu gewähren und um Probleme mit Zugriffsberechtigungen für einige AWS Proton Konsolenoperationen zu beheben.	30. März 2022
AWSProtonReadOnlyAccess – Aktualisierung auf eine bestehende Richtlinie	AWS Proton aktualisieren Sie diese Richtlinie, um Zugriff auf neue AWS Proton API-Operationen zu gewähren und um Berechtigungsprobleme bei einigen AWS Proton Konsolenoperationen zu beheben.	30. März 2022

Änderungen	Beschreibung	Datum
AWSProtonSyncServiceRolePolicy – Neue Richtlinie	AWS Proton hat eine neue Richtlinie hinzugefügt, die die Ausführung von Vorgängen im Zusammenhang mit der Vorlagensynchronisierung ermöglicht AWS Proton . Die Richtlinie wird in der AWSServiceRoleForProtonSyncserviceverknüpften Rolle verwendet.	23. November 2021
AWSProtonFullAccess – Neue Richtlinie	AWS Proton hat eine neue Richtlinie hinzugefügt, um Administratorrollen Zugriff auf AWS Proton API-Operationen und die AWS Proton Konsole zu gewähren.	09. Juni 2021
AWSProtonDeveloperAccess – Neue Richtlinie	AWS Proton Es wurde eine neue Richtlinie hinzugefügt, um Entwicklerrollen Zugriff auf AWS Proton API-Operationen und die AWS Proton Konsole zu gewähren.	09. Juni 2021
AWSProtonReadOnlyAccess – Neue Richtlinie	AWS Proton hat eine neue Richtlinie hinzugefügt, um schreibgeschützten Zugriff auf AWS Proton API-Operationen und die AWS Proton Konsole zu gewähren.	09. Juni 2021

Änderungen	Beschreibung	Datum
AWS Proton hat begonnen, Änderungen zu verfolgen.	AWS Proton hat begonnen, Änderungen für die AWS verwalteten Richtlinien zu verfolgen.	09. Juni 2021

Verwenden von serviceverknüpften Rollen für AWS Proton

AWS Proton verwendet AWS Identity and Access Management (IAM) serviceverknüpfte Rollen. Eine serviceverknüpfte Rolle ist ein einzigartiger Typ von IAM-Rolle, mit der direkt verknüpft ist. AWS Proton Mit Diensten verknüpfte Rollen sind vordefiniert AWS Proton und enthalten alle Berechtigungen, die der Dienst benötigt, um andere AWS Dienste in Ihrem Namen aufzurufen.

Themen

- Rollen für AWS Proton die Synchronisierung verwenden
- Verwendung von Rollen für die CodeBuild basierte Bereitstellung

Rollen für AWS Proton die Synchronisierung verwenden

AWS Proton verwendet AWS Identity and Access Management (IAM) serviceverknüpfte Rollen. Eine serviceverknüpfte Rolle ist ein einzigartiger Typ von IAM-Rolle, mit der direkt verknüpft ist. AWS Proton Mit Diensten verknüpfte Rollen sind vordefiniert AWS Proton und enthalten alle Berechtigungen, die der Dienst benötigt, um andere AWS Dienste in Ihrem Namen aufzurufen.

Eine dienstbezogene Rolle AWS Proton erleichtert die Einrichtung, da Sie die erforderlichen Berechtigungen nicht manuell hinzufügen müssen. AWS Proton definiert die Berechtigungen ihrer dienstbezogenen Rollen und AWS Proton kann, sofern nicht anders definiert, nur ihre Rollen übernehmen. Die definierten Berechtigungen umfassen die Vertrauens- und Berechtigungsrichtlinie. Diese Berechtigungsrichtlinie kann keinen anderen IAM-Entitäten zugewiesen werden.

Sie können eine serviceverknüpfte Rolle erst löschen, nachdem ihre verwandten Ressourcen gelöscht wurden. Dadurch werden Ihre AWS Proton Ressourcen geschützt, da Sie nicht versehentlich die Zugriffsberechtigung für die Ressourcen entziehen können.

Informationen zu anderen Diensten, die dienstverknüpfte Rollen unterstützen, finden Sie unter AWS Dienste, die mit IAM funktionieren. Suchen Sie in der Spalte Dienstverknüpfte Rollen nach

den Diensten, für die Ja steht. Wählen Sie über einen Link Ja aus, um die Dokumentation zu einer serviceverknüpften Rolle für diesen Service anzuzeigen.

Berechtigungen für dienstverknüpfte Rollen AWS Proton

AWS Proton verwendet zwei dienstbezogene Rollen mit dem Namen AWSServiceRoleForProtonSyncund. AWSServiceRoleForProtonServiceSync

Die serviceverknüpfte Rolle AWSServiceRoleForProtonSync vertraut darauf, dass die folgenden Services die Rolle annehmen:

- `sync.proton.amazonaws.com`

Die genannte Rollenberechtigungsrichtlinie `AWSProtonSyncServiceRolePolicy` AWS Proton ermöglicht es, die folgenden Aktionen für die angegebenen Ressourcen durchzuführen:

- Aktion: AWS Proton Vorlagen und Vorlagenversionen erstellen, verwalten und weiterlesen
- Aktion: Verbindung verwenden am CodeConnections

Weitere Informationen zu dieser Richtlinie finden Sie unter AWS verwaltete Richtlinie: AWSProton SyncServiceRolePolicy.

Die serviceverknüpfte Rolle AWSServiceRoleForProtonServiceSync vertraut darauf, dass die folgenden Services die Rolle annehmen:

- `service-sync.proton.amazonaws.com`

Die genannte Rollenberechtigungsrichtlinie `AWSProtonServiceGitSyncServiceRolePolicy` AWS Proton ermöglicht es, die folgenden Aktionen für die angegebenen Ressourcen durchzuführen:

- Aktion: AWS Proton Dienste und Dienstinstanzen erstellen, verwalten und darauf lesen

Weitere Informationen zu dieser Richtlinie finden Sie unter AWS verwaltete Richtlinie: AWSProton ServiceGitSyncServiceRolePolicy.

Sie müssen Berechtigungen konfigurieren, damit eine juristische Stelle von IAM (z. B. Benutzer, Gruppe oder Rolle) eine serviceverknüpfte Rolle erstellen, bearbeiten oder löschen kann. Weitere Informationen finden Sie unter Serviceverknüpfte Rollenberechtigung im IAM-Benutzerhandbuch.

Erstellen einer serviceverknüpften Rolle für AWS Proton

Sie müssen eine serviceverknüpfte Rolle nicht manuell erstellen. Wenn Sie ein Repository oder einen Dienst für die Synchronisierung AWS Proton in der AWS-Managementkonsole, der oder der AWS API konfigurieren AWS CLI, AWS Proton erstellt die dienstverknüpfte Rolle für Sie.

Wenn Sie diese serviceverknüpfte Rolle löschen und sie dann erneut erstellen müssen, können Sie dasselbe Verfahren anwenden, um die Rolle in Ihrem Konto neu anzulegen. Wenn Sie ein Repository oder einen Dienst für die Synchronisierung konfigurieren AWS Proton, AWS Proton erstellt die dienstverknüpfte Rolle erneut für Sie.

Um die AWSServiceRoleForProtonSyncdienstverknüpfte Rolle neu zu erstellen, sollten Sie ein Repository für die Synchronisierung konfigurieren, und für die AWSServiceRoleForProtonServiceSyncNeuerstellung sollten Sie einen Dienst für die Synchronisierung konfigurieren.

Eine dienstverknüpfte Rolle bearbeiten für AWS Proton

AWS Proton erlaubt es Ihnen nicht, die AWSServiceRoleForProtonSyncdienstbezogene Rolle zu bearbeiten. Da möglicherweise verschiedene Entitäten auf die Rolle verweisen, kann der Rollenname nach der Erstellung einer serviceverknüpften Rolle nicht bearbeitet werden. Sie können jedoch die Beschreibung der Rolle mit IAM bearbeiten. Weitere Informationen finden Sie unter Bearbeiten einer serviceverknüpften Rolle im IAM-Benutzerhandbuch.

Löschen einer dienstbezogenen Rolle für AWS Proton

Sie müssen die Rolle AWSServiceRoleForProtonSync nicht manuell löschen. Wenn Sie alle AWS Proton verknüpften Repositorys für die Repository-Synchronisierung in der AWS-Managementkonsole AWS CLI, der oder der AWS API löschen, werden die Ressourcen AWS Proton bereinigt und die dienstverknüpfte Rolle für Sie gelöscht.

Unterstützte Regionen für dienstverknüpfte Rollen AWS Proton

AWS Proton unterstützt die Verwendung von dienstbezogenen Rollen überall dort, AWS-Regionen wo der Dienst verfügbar ist. Weitere Informationen finden Sie unter AWS Proton -Endpunkte und -Kontingente in der Allgemeine AWS-Referenz.

Verwendung von Rollen für die CodeBuild basierte Bereitstellung

AWS Proton verwendet AWS Identity and Access Management (IAM) serviceverknüpfte Rollen. Eine serviceverknüpfte Rolle ist ein einzigartiger Typ von IAM-Rolle, mit der direkt verknüpft ist.

AWS Proton Mit Diensten verknüpfte Rollen sind vordefiniert AWS Proton und enthalten alle Berechtigungen, die der Dienst benötigt, um andere AWS Dienste in Ihrem Namen aufzurufen.

Eine dienstbezogene Rolle AWS Proton erleichtert die Einrichtung, da Sie die erforderlichen Berechtigungen nicht manuell hinzufügen müssen. AWS Proton definiert die Berechtigungen ihrer dienstbezogenen Rollen und AWS Proton kann, sofern nicht anders definiert, nur ihre Rollen übernehmen. Die definierten Berechtigungen umfassen die Vertrauens- und Berechtigungsrichtlinie. Diese Berechtigungsrichtlinie kann keinen anderen IAM-Entitäten zugewiesen werden.

Sie können eine serviceverknüpfte Rolle erst löschen, nachdem ihre verwandten Ressourcen gelöscht wurden. Dadurch werden Ihre AWS Proton Ressourcen geschützt, da Sie nicht versehentlich die Zugriffsberechtigung für die Ressourcen entziehen können.

Informationen zu anderen Diensten, die dienstverknüpfte Rollen unterstützen, finden Sie unter AWS Dienste, die mit IAM funktionieren. Suchen Sie in der Spalte Dienstverknüpfte Rollen nach den Diensten, für die Ja steht. Wählen Sie über einen Link Ja aus, um die Dokumentation zu einer serviceverknüpften Rolle für diesen Service anzuzeigen.

Berechtigungen für dienstverknüpfte Rollen AWS Proton

AWS Proton verwendet die dienstverknüpfte Rolle mit dem Namen AWSServiceRoleForProtonCodeBuildProvisioning— Eine dienstverknüpfte Rolle für die AWS Proton CodeBuild Bereitstellung.

Die serviceverknüpfte Rolle AWSServiceRoleForProtonCodeBuildProvisioning vertraut darauf, dass die folgenden Services die Rolle annehmen:

- `codebuild.proton.amazonaws.com`

Die genannte Richtlinie für Rollenberechtigungen `AWSProtonCodeBuildProvisioningServiceRolePolicy` AWS Proton ermöglicht es, die folgenden Aktionen für die angegebenen Ressourcen durchzuführen:

- Aktion: CloudFormation Stacks und Transformationen erstellen, verwalten und darauf lesen
- Aktion: CodeBuild Projekte und Builds erstellen, verwalten und weiterlesen

Weitere Informationen zu dieser Richtlinie finden Sie unter AWS verwaltete Richtlinie: AWSProton CodeBuildProvisioningServiceRolePolicy.

Sie müssen Berechtigungen konfigurieren, damit eine juristische Stelle von IAM (z. B. Benutzer, Gruppe oder Rolle) eine serviceverknüpfte Rolle erstellen, bearbeiten oder löschen kann. Weitere Informationen finden Sie unter Serviceverknüpfte Rollenberechtigung im IAM-Benutzerhandbuch.

Erstellen einer serviceverknüpften Rolle für AWS Proton

Sie müssen eine serviceverknüpfte Rolle nicht manuell erstellen. Wenn Sie eine Umgebung erstellen, die CodeBuild basiertes Provisioning AWS Proton in der AWS-Managementkonsole, der oder der AWS API verwendet AWS CLI, AWS Proton wird die serviceverknüpfte Rolle für Sie erstellt.

Wenn Sie diese serviceverknüpfte Rolle löschen und sie dann erneut erstellen müssen, können Sie dasselbe Verfahren anwenden, um die Rolle in Ihrem Konto neu anzulegen. Wenn Sie eine Umgebung erstellen, in der CodeBuild basiertes Provisioning verwendet AWS Proton wird AWS Proton, wird die serviceverknüpfte Rolle erneut für Sie erstellt.

Bearbeiten einer serviceverknüpften Rolle für AWS Proton

AWS Proton erlaubt es Ihnen nicht, die AWSServiceRoleForProtonCodeBuildProvisioningdienstbezogene Rolle zu bearbeiten. Da möglicherweise verschiedene Entitäten auf die Rolle verweisen, kann der Rollenname nach dem Erstellen einer serviceverknüpften Rolle nicht mehr geändert werden. Sie können jedoch die Beschreibung der Rolle mit IAM bearbeiten. Weitere Informationen finden Sie unter Bearbeiten einer serviceverknüpften Rolle im IAM-Benutzerhandbuch.

Löschen einer dienstbezogenen Rolle für AWS Proton

Wenn Sie ein Feature oder einen Dienst, die bzw. der eine serviceverknüpften Rolle erfordert, nicht mehr benötigen, sollten Sie diese Rolle löschen. Auf diese Weise haben Sie keine ungenutzte juristische Stelle, die nicht aktiv überwacht oder verwaltet wird. Sie müssen jedoch alle Umgebungen und Dienste (Instanzen und Pipelines) löschen, die CodeBuild Based Provisioning verwenden, AWS Proton bevor Sie sie manuell löschen können.

Manuelles Löschen der serviceverknüpften Rolle

Verwenden Sie die IAM-Konsole, die oder die AWS API AWS CLI, um die serviceverknüpfte Rolle zu löschen. AWSServiceRoleForProtonCodeBuildProvisioning Weitere Informationen finden Sie unter Löschen einer serviceverknüpften Rolle im IAM-Benutzerhandbuch.

Unterstützte Regionen für serviceverknüpfte Rollen AWS Proton

AWS Proton unterstützt die Verwendung von dienstbezogenen Rollen überall dort, AWS-Regionen wo der Dienst verfügbar ist. Weitere Informationen finden Sie unter AWS Proton -Endpunkte und -Kontingente in der Allgemeine AWS-Referenz.

Problembehandlung bei AWS Proton Identität und Zugriff

Verwenden Sie die folgenden Informationen, um häufig auftretende Probleme zu diagnostizieren und zu beheben, die bei der Arbeit mit AWS Proton und IAM auftreten können.

Themen

- Ich bin nicht berechtigt, eine Aktion durchzuführen in AWS Proton
- Ich bin nicht berechtigt, iam auszuführen: PassRole
- Ich möchte Personen außerhalb von mir den Zugriff AWS-Konto auf meine AWS Proton Ressourcen ermöglichen

Ich bin nicht berechtigt, eine Aktion durchzuführen in AWS Proton

Wenn Ihnen AWS-Managementkonsole mitgeteilt wird, dass Sie nicht berechtigt sind, eine Aktion durchzuführen, müssen Sie sich an Ihren Administrator wenden, um Unterstützung zu erhalten. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Der folgende Beispielfehler tritt auf, wenn der `mateojackson` IAM-Benutzer versucht, die Konsole zum Anzeigen von Details zu einer fiktiven *`my-example-widget`*-Ressource zu verwenden, jedoch nicht über `proton:`*`GetWidget`*-Berechtigungen verfügt.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 proton:GetWidget on resource: my-example-widget
```

In diesem Fall bittet Mateo seinen Administrator um die Aktualisierung seiner Richtlinien, um unter Verwendung der Aktion *`my-example-widget`* auf die Ressource `proton:`*`GetWidget`* zugreifen zu können.

Ich bin nicht berechtigt, iam auszuführen: PassRole

Wenn Sie die Fehlermeldung erhalten, dass Sie nicht zum Durchführen der `iam:PassRole`-Aktion autorisiert sind, müssen Ihre Richtlinien aktualisiert werden, um eine Rolle an AWS Protonübergeben zu können.

Einige AWS-Services ermöglichen es Ihnen, eine bestehende Rolle an diesen Dienst zu übergeben, anstatt eine neue Servicerolle oder eine dienstverknüpfte Rolle zu erstellen. Hierzu benötigen Sie Berechtigungen für die Übergabe der Rolle an den Dienst.

Der folgende Beispielfehler tritt auf, wenn ein IAM-Benutzer mit dem Namen `marymajor` versucht, die Konsole zu verwenden, um eine Aktion in AWS Proton auszuführen. Die Aktion erfordert jedoch, dass der Service über Berechtigungen verfügt, die durch eine Servicerolle gewährt werden. Mary besitzt keine Berechtigungen für die Übergabe der Rolle an den Dienst.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In diesem Fall müssen die Richtlinien von Mary aktualisiert werden, um die Aktion `iam:PassRole` ausführen zu können.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich möchte Personen außerhalb von mir den Zugriff AWS-Konto auf meine AWS Proton Ressourcen ermöglichen

Sie können eine Rolle erstellen, mit der Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation auf Ihre Ressourcen zugreifen können. Sie können festlegen, wem die Übernahme der Rolle anvertraut wird. Für Dienste, die ressourcenbasierte Richtlinien oder Zugriffskontrolllisten (ACLs) unterstützen, können Sie diese Richtlinien verwenden, um Personen Zugriff auf Ihre Ressourcen zu gewähren.

Weitere Informationen dazu finden Sie hier:

- Informationen darüber, ob diese Funktionen AWS Proton unterstützt werden, finden Sie unter. Wie AWS Proton funktioniert mit IAM
- Informationen dazu, wie Sie Zugriff auf Ihre Ressourcen gewähren können, AWS-Konten die Ihnen gehören, finden Sie im IAM-Benutzerhandbuch unter Gewähren des Zugriffs auf einen IAM-Benutzer in einem anderen AWS-Konto , den Sie besitzen.
- Informationen dazu, wie Sie Dritten Zugriff auf Ihre Ressourcen gewähren können AWS-Konten, finden Sie AWS-Konten im IAM-Benutzerhandbuch unter Gewähren des Zugriffs für Dritte.
- Informationen dazu, wie Sie über einen Identitätsverbund Zugriff gewähren, finden Sie unter Gewähren von Zugriff für extern authentifizierte Benutzer (Identitätsverbund) im IAM-Benutzerhandbuch.

- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

Konfiguration und Schwachstellenanalyse in AWS Proton

AWS Proton stellt keine Patches oder Updates für vom Kunden bereitgestellten Code bereit. Kunden sind dafür verantwortlich, ihren eigenen Code zu aktualisieren und Patches anzubringen, einschließlich des Quellcodes für ihre Dienste und Anwendungen, auf denen sie ausgeführt werden, AWS Proton und des Codes, der in ihren Service- und Umgebungsvorlagenpaketen bereitgestellt wird.

Kunden sind dafür verantwortlich, die Infrastrukturressourcen in ihren Umgebungen und Diensten zu aktualisieren und zu patchen. AWS Proton aktualisiert oder patcht keine Ressourcen automatisch. Kunden sollten die Dokumentation zu den Ressourcen in ihrer Architektur lesen, um ihre jeweiligen Patching-Richtlinien zu verstehen.

Abgesehen von der Bereitstellung von vom Kunden angeforderten Umgebungs- und Service-Updates für Nebenversionen von Service- und Umgebungsvorlagen werden AWS Proton keine Patches oder Updates für die Ressourcen bereitgestellt, die Kunden in ihren Service- und Umgebungsvorlagen und Vorlagenpaketen definieren.

Weitere Informationen finden Sie in den folgenden Ressourcen:

- Modell der übergreifenden Verantwortlichkeit
- Amazon Web Services – Übersicht über Sicherheitsverfahren

Datenschutz in AWS Proton

AWS Proton entspricht dem Modell der AWS gemeinsamen Verantwortung mit geteilter der , das Vorschriften und Richtlinien für den Datenschutz enthält. AWS ist verantwortlich für den Schutz der globalen Infrastruktur, die AWS-Services alle betreibt. AWS behält die Kontrolle über die auf dieser Infrastruktur gehosteten Daten, einschließlich der Sicherheitskonfigurationskontrollen für den Umgang mit Kundeninhalten und personenbezogenen Daten. AWS Kunden und APN-Partner, die entweder als Datenverantwortliche oder als Datenverarbeiter agieren, sind für alle personenbezogenen Daten verantwortlich, die sie in die AWS Cloud

Aus Datenschutzgründen empfehlen wir Ihnen, Ihre AWS-Konto Anmeldeinformationen zu schützen und individuelle Benutzerkonten bei AWS Identity and Access Management (IAM) einzurichten, sodass jeder Benutzer nur die Berechtigungen erhält, die für die Erfüllung seiner Aufgaben erforderlich sind. Außerdem empfehlen wir, die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die Multi-Faktor-Authentifizierung (MFA).
- Verwenden Sie SSL/TLS, um mit Ressourcen zu kommunizieren. AWS Wir empfehlen TLS 1.2 oder höher.
- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit ein. AWS CloudTrail
- Verwenden Sie AWS Verschlüsselungslösungen zusammen mit allen darin enthaltenen Standardsicherheitskontrollen AWS-Services.

Wir empfehlen dringend, niemals vertrauliche Identifikationsinformationen, wie z. B. die Kontonummern Ihrer Kunden, in frei formatierte Textfelder wie ein Namensfeld einzugeben. Dies gilt auch, wenn Sie mit der Konsole, der AWS-Services API AWS Proton oder auf andere Weise arbeiten oder AWS SDKs diese verwenden. AWS CLI Alle Daten, die Sie in Freiform-Textfelder für Ressourcen-IDs oder ähnliche Elemente im Zusammenhang mit der Verwaltung von AWS Ressourcen eingeben, können für die Aufnahme in Diagnoseprotokolle aufgenommen werden. Wenn Sie eine URL für einen externen Server bereitstellen, schließen Sie keine Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL ein.

Weitere Informationen zum Datenschutz enthält der Blog-Beitrag AWS Shared Responsibility Model and GDPR im AWS -Sicherheitsblog.

Serverseitige Verschlüsselung im Ruhezustand

Wenn Sie sich dafür entscheiden, vertrauliche Daten in Ihren gespeicherten Vorlagenpaketen im S3-Bucket zu verschlüsseln, in dem Sie Ihre Vorlagenpakete speichern, müssen Sie einen SSE-S3- oder SSE-KMS-Schlüssel verwenden, um das Abrufen der Vorlagenpakete AWS Proton zu ermöglichen, sodass sie an eine registrierte Vorlage angehängt werden können. AWS Proton

Verschlüsselung während der Übertragung

Die gesamte Service-zu-Service-Kommunikation wird während der Übertragung mit SSL/TLS verschlüsselt.

AWS Proton Verwaltung von Verschlüsselungsschlüsseln

Darin AWS Proton werden alle Kundendaten standardmäßig mit einem AWS Proton eigenen Schlüssel verschlüsselt. Wenn Sie einen kundeneigenen und verwalteten AWS KMS Schlüssel angeben, werden alle Kundendaten mit dem vom Kunden bereitgestellten Schlüssel verschlüsselt, wie in den folgenden Abschnitten beschrieben.

Wenn Sie eine AWS Proton Vorlage erstellen, geben Sie Ihren Schlüssel an und AWS Proton verwenden Ihre Anmeldeinformationen, um ein Zertifikat zu erstellen, das die Verwendung Ihres Schlüssels ermöglicht AWS Proton .

Wenn Sie den Grant manuell zurückziehen oder Ihren angegebenen Schlüssel deaktivieren oder löschen AWS Proton , können die Daten, die mit dem angegebenen Schlüssel verschlüsselt wurden, nicht gelesen werden. `ValidationException`

AWS Proton Verschlüsselungskontext

AWS Proton unterstützt Header für den Verschlüsselungskontext. Ein Verschlüsselungskontext ist ein optionaler Satz von Schlüssel-Wert-Paaren, die zusätzliche kontextbezogene Informationen zu den Daten enthalten können. Allgemeine Informationen zum Verschlüsselungs-Kontext finden Sie unter AWS Key Management Service Concepts – Encryption Context (Konzepte – Verschlüsselungs-Kontext) im AWS Key Management Service -Entwicklerhandbuch.

Ein Verschlüsselungskontext ist ein Satz von Schlüssel-Wert-Paaren, die beliebige, nicht geheime Daten enthalten. Wenn ein Verschlüsselungskontext in eine Anfrage zur Verschlüsselung von Daten aufgenommen wird, wird der Verschlüsselungskontext AWS KMS kryptografisch an die verschlüsselten Daten gebunden. Zur Entschlüsselung der Daten müssen Sie denselben Verschlüsselungskontext übergeben.

Kunden können den Verschlüsselungskontext verwenden, um die Verwendung ihres vom Kunden verwalteten Schlüssels in Prüfaufzeichnungen und Protokollen nachzuweisen. Es erscheint auch im Klartext in Protokollen wie AWS CloudTrail Amazon CloudWatch Logs.

AWS Proton berücksichtigt keinen vom Kunden oder extern spezifizierten Verschlüsselungskontext.

AWS Proton fügt den folgenden Verschlüsselungskontext hinzu.

```
{
  "aws:proton:template": "<proton-template-arn>",
  "aws:proton:resource": "<proton-resource-arn>"
```

```
}
```

Der erste Verschlüsselungskontext identifiziert die AWS Proton Vorlage, der die Ressource zugeordnet ist, und dient auch als Einschränkung für vom Kunden verwaltete Schlüsselberechtigungen und -erteilungen.

Der zweite Verschlüsselungskontext identifiziert die AWS Proton Ressource, die verschlüsselt ist.

Die folgenden Beispiele zeigen die Verwendung des AWS Proton Verschlüsselungskontextes.

Entwickler, der eine Dienstinstanz erstellt.

```
{
  "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"
}
```

Ein Administrator, der eine Vorlage erstellt.

```
{
  "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"
}
```

Infrastruktursicherheit in AWS Proton

Als verwalteter Dienst AWS Proton ist er durch AWS globale Netzwerksicherheit geschützt. Informationen zu AWS Sicherheitsdiensten und zum AWS Schutz der Infrastruktur finden Sie unter AWS Cloud-Sicherheit. Informationen zum Entwerfen Ihrer AWS Umgebung unter Verwendung der bewährten Methoden für die Infrastruktursicherheit finden Sie unter Infrastructure Protection in Security Pillar AWS Well-Architected Framework.

Sie verwenden AWS veröffentlichte API-Aufrufe für den Zugriff AWS Proton über das Netzwerk. Kunden müssen Folgendes unterstützen:

- Transport Layer Security (TLS). Wir benötigen TLS 1.2 und empfehlen TLS 1.3.

- Verschlüsselungs-Suiten mit Perfect Forward Secrecy (PFS) wie DHE (Ephemeral Diffie-Hellman) oder ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Die meisten modernen Systeme wie Java 7 und höher unterstützen diese Modi.

Um die Netzwerkisolierung zu verbessern, können Sie die im folgenden Abschnitt beschriebene Methode verwenden AWS PrivateLink .

AWS Proton und Schnittstellen-VPC-Endpunkte ()AWS PrivateLink

Sie können eine private Verbindung zwischen Ihrer VPC herstellen und AWS Proton einen VPC-Schnittstellen-Endpunkt erstellen. Schnittstellenendpunkte werden mit einer Technologie betrieben AWS PrivateLink, die Ihnen den privaten Zugriff AWS Proton APIs ohne Internet-Gateway, NAT-Gerät, VPN-Verbindung oder Verbindung ermöglicht. AWS Direct Connect Instances in Ihrer VPC benötigen keine öffentlichen IP-Adressen für die Kommunikation. AWS Proton APIs Datenverkehr zwischen Ihrer VPC und AWS Proton verlässt das Amazon-Netzwerk nicht.

Jeder Schnittstellenendpunkt wird durch eine oder mehrere Elastic-Network-Schnittstellen in Ihren Subnetzen dargestellt.

Weitere Informationen finden Sie unter Schnittstellen-VPC-Endpunkte (AWS PrivateLink) im Amazon-VPC-Benutzerhandbuch.

Überlegungen zu AWS Proton VPC-Endpunkten

Bevor Sie einen Schnittstellen-VPC-Endpunkt für einrichten, stellen Sie sicher AWS Proton, dass Sie die Eigenschaften und Einschränkungen der Schnittstellen-Endpunkte im Amazon VPC-Benutzerhandbuch lesen.

AWS Proton unterstützt Aufrufe aller API-Aktionen von Ihrer VPC aus.

VPC-Endpunktrichtlinien werden unterstützt für AWS Proton. Standardmäßig AWS Proton ist der vollständige Zugriff auf über den Endpunkt zulässig. Weitere Informationen finden Sie unter Steuerung des Zugriffs auf Services mit VPC-Endpunkten im Amazon-VPC-Benutzerhandbuch.

Erstellen eines Schnittstellen-VPC-Endpunkts für AWS Proton

Sie können einen VPC-Endpunkt für den AWS Proton Service entweder mit der Amazon VPC-Konsole oder mit AWS Command Line Interface ()AWS CLI erstellen. Weitere Informationen finden Sie unter Erstellung eines Schnittstellenendpunkts im Benutzerhandbuch für Amazon VPC.

Erstellen Sie einen VPC-Endpunkt für die AWS Proton Verwendung des folgenden Dienstnamens:

- com.amazonaws. *region*. Proton

Wenn Sie privates DNS für den Endpunkt aktivieren, können Sie API-Anfragen an die AWS Proton Verwendung des Standard-DNS-Namens für die Region stellen, z. B. `proton.region.amazonaws.com`

Weitere Informationen finden Sie unter Zugriff auf einen Service über einen Schnittstellenendpunkt im Benutzerhandbuch für Amazon VPC.

Erstellen einer VPC-Endpunktrichtlinie für AWS Proton

Sie können eine Endpunktrichtlinie an Ihren VPC-Endpunkt anhängen, der den Zugriff auf AWS Proton steuert. Die Richtlinie gibt die folgenden Informationen an:

- Prinzipal, der die Aktionen ausführen kann.
- Aktionen, die ausgeführt werden können
- Die Ressourcen, für die Aktionen ausgeführt werden können.

Weitere Informationen finden Sie unter Steuerung des Zugriffs auf Services mit VPC-Endpunkten im Amazon-VPC-Benutzerhandbuch.

Beispiel: VPC-Endpunktrichtlinie für Aktionen AWS Proton

Im Folgenden finden Sie ein Beispiel für eine Endpunktrichtlinie für AWS Proton. Wenn diese Richtlinie an einen Endpunkt angehängt ist, gewährt sie allen Prinzipalen auf allen Ressourcen Zugriff auf die aufgelisteten AWS Proton Aktionen.

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": "*"
    }
  ]
}
```

Einloggen und Überwachen AWS Proton

Die Überwachung ist ein wichtiger Bestandteil der Aufrechterhaltung der Zuverlässigkeit, Verfügbarkeit und Leistung Ihrer AWS Proton anderen AWS Lösungen. AWS bietet die folgenden Überwachungstools, mit denen Sie Ihre Instances beobachten AWS Proton, melden können, wenn etwas nicht stimmt, und bei Bedarf automatische Maßnahmen ergreifen können.

Derzeit ist es AWS Proton nicht in Amazon CloudWatch Logs oder integriert AWS Trusted Advisor. Administratoren können andere konfigurieren und CloudWatch zur Überwachung verwenden, AWS-Services wie in ihren Service- und Umgebungsvorlagen definiert. AWS Proton ist integriert in AWS CloudTrail.

- Amazon CloudWatch überwacht Ihre AWS Ressourcen und die Anwendungen, auf denen Sie laufen, AWS in Echtzeit. Sie können Kennzahlen erfassen und verfolgen, benutzerdefinierte Dashboards erstellen und Alarme festlegen, die Sie benachrichtigen oder Maßnahmen ergreifen, wenn eine bestimmte Metrik einen von Ihnen festgelegten Schwellenwert erreicht. Sie können beispielsweise die CPU-Auslastung oder andere Kennzahlen Ihrer EC2 Amazon-Instances CloudWatch verfolgen und bei Bedarf automatisch neue Instances starten. Weitere Informationen finden Sie im CloudWatch Amazon-Benutzerhandbuch.

- Mit Amazon CloudWatch Logs können Sie Ihre Protokolldateien von EC2 Amazon-Instances und anderen Quellen überwachen CloudTrail, speichern und darauf zugreifen. CloudWatch Logs kann Informationen in den Protokolldateien überwachen und Sie benachrichtigen, wenn bestimmte Schwellenwerte erreicht werden. Sie können Ihre Protokolldaten auch in einem sehr robusten Speicher archivieren. Weitere Informationen finden Sie im Amazon CloudWatch Logs-Benutzerhandbuch.
- AWS CloudTrailerfasst API-Aufrufe und zugehörige Ereignisse, die von Ihnen oder in Ihrem Namen getätigt wurden, AWS-Konto und übermittelt die Protokolldateien an einen von Ihnen angegebenen Amazon S3 S3-Bucket. Sie können feststellen, welche Benutzer und Konten angerufen wurden AWS, von welcher Quell-IP-Adresse aus die Anrufe getätigt wurden und wann die Aufrufe erfolgten. Weitere Informationen finden Sie im AWS CloudTrail -Benutzerhandbuch.
- Amazon EventBridge ist ein serverloser Event-Bus-Service, der es einfach macht, Ihre Anwendungen mit Daten aus einer Vielzahl von Quellen zu verbinden. EventBridge liefert einen Stream von Echtzeitdaten aus Ihren eigenen Anwendungen Software-as-a-Service (SaaS) AWS-Services und leitet diese Daten an Ziele wie Lambda weiter. Auf diese Weise können Sie Ereignisse überwachen, die in Services auftreten, und ereignisgesteuerte Architekturen erstellen. Weitere Informationen finden Sie im Automatisieren Sie AWS Proton mit EventBridge und dem EventBridge -Benutzerhandbuch.

Resilienz in AWS Proton

Die AWS globale Infrastruktur basiert auf Availability AWS-Region Zones. AWS-Region s bieten mehrere physisch getrennte und isolierte Availability Zones, die über Netzwerke mit niedriger Latenz, hohem Durchsatz und hoher Redundanz miteinander verbunden sind. Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Zonen ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen zu AWS-Region s und Availability Zones finden Sie unter AWS Globale Infrastruktur.

Zusätzlich zur AWS globalen Infrastruktur AWS Proton bietet es Funktionen zur Unterstützung Ihrer Datenausfallsicherheit und Backup-Anforderungen.

AWS Proton Backups

AWS Proton verwaltet eine Sicherungskopie aller Kundendaten. Im Falle eines Totalausfalls kann dieses Backup verwendet werden, um Kundendaten aus einem früheren gültigen Zustand wiederherzustellen AWS Proton .

Bewährte Sicherheitsmethoden für AWS Proton

AWS Proton enthält Sicherheitsfunktionen, die Sie bei der Entwicklung und Implementierung Ihrer eigenen Sicherheitsrichtlinien berücksichtigen sollten. Die folgenden bewährten Methoden sind allgemeine Richtlinien und keine vollständige Sicherheitslösung. Da diese bewährten Methoden für Ihre Umgebung möglicherweise nicht angemessen oder ausreichend sind, sollten Sie sie als hilfreiche Überlegungen und nicht als bindend ansehen.

Themen

Verwendung von IAM für die Zugriffskontrolle

IAM ist ein AWS-Service Programm, mit dem Sie Benutzer und deren Berechtigungen verwalten können. AWS Sie können IAM with verwenden AWS Proton , um festzulegen, welche AWS Proton Aktionen Administratoren und Entwickler ausführen können, z. B. die Verwaltung von Vorlagen, Umgebungen oder Diensten. Sie können IAM-Dienstrollen verwenden, um in Ihrem Namen Anrufe an andere Dienste zu tätigen. AWS Proton

Weitere Informationen zu AWS Proton und IAM-Rollen finden Sie unter. Identity and Access Management für AWS Proton

Implementieren Sie den Zugriff mit den geringsten Rechten. Weitere Informationen finden Sie im AWS Identity and Access Management Benutzerhandbuch unter Richtlinien und Berechtigungen in IAM.

Betten Sie keine Anmeldeinformationen in Ihre Vorlagen und Vorlagenpakete ein

Anstatt vertrauliche Informationen in Ihre CloudFormation Vorlagen und Vorlagenpakete einzubetten, empfehlen wir Ihnen, dynamische Verweise in Ihrer Stack-Vorlage zu verwenden.

Dynamische Verweise bieten Ihnen eine kompakte und leistungsstarke Möglichkeit, auf externe Werte zu verweisen, die in anderen Diensten gespeichert und verwaltet werden, z. B. im AWS Systems Manager Parameter Store oder. AWS Secrets Manager Wenn Sie eine dynamische Referenz verwenden, CloudFormation ruft sie bei Bedarf bei Stapel- und Änderungssatzoperationen den Wert der angegebenen Referenz ab und übergibt den Wert an die entsprechende Ressource. Speichert jedoch CloudFormation niemals den tatsächlichen Referenzwert. Weitere Informationen finden Sie im CloudFormation Benutzerhandbuch unter Verwenden dynamischer Referenzen zur Angabe von Vorlagenwerten.

AWS Secrets Managerhilft Ihnen dabei, Anmeldeinformationen für Ihre Datenbanken und andere Dienste sicher zu verschlüsseln, zu speichern und abzurufen. Der AWS Systems Manager Parameter Store bietet sicheren, hierarchischen Speicher für die Verwaltung von Konfigurationsdaten.

Weitere Informationen zur Definition von Vorlagenparametern finden Sie https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/parameters-section-structure.html im CloudFormation Benutzerhandbuch.

Verwenden Sie Verschlüsselung, um vertrauliche Daten zu schützen

Darin AWS Proton werden alle Kundendaten standardmäßig mit einem AWS Proton eigenen Schlüssel verschlüsselt.

Als Mitglied des Plattformteams können Sie einen vom Kunden verwalteten Schlüssel bereitstellen, um Ihre sensiblen Daten AWS Proton zu verschlüsseln und zu sichern. Verschlüsseln Sie sensible Daten, die sich in Ihrem S3-Bucket befinden. Weitere Informationen finden Sie unter Datenschutz in AWS Proton.

Wird AWS CloudTrail zum Anzeigen und Protokollieren von API-Aufrufen verwendet

AWS CloudTrail verfolgt jeden, der API-Aufrufe in Ihrem macht AWS-Konto. API-Aufrufe werden protokolliert, wenn jemand die AWS Proton API, die AWS Proton Konsole oder AWS Proton AWS

CLI Befehle verwendet. Aktivieren Sie die Protokollierung und legen Sie einen Amazon-S3-Bucket zum Speichern der Protokolle fest. Auf diese Weise können Sie bei Bedarf überprüfen, wer welchen AWS Proton Anruf in Ihrem Konto getätigt hat. Weitere Informationen finden Sie unter Einloggen und Überwachen AWS Proton.

Serviceübergreifende Confused-Deputy-Prävention

Das Confused-Deputy-Problem ist ein Sicherheitsproblem, bei dem eine juristische Stelle, die nicht über die Berechtigung zum Ausführen einer Aktion verfügt, eine privilegiertere juristische Stelle zwingen kann, die Aktion auszuführen. In AWS, dienstübergreifender Identitätswechsel kann zum Problem des verwirrten Stellvertreters führen. Ein dienstübergreifender Identitätswechsel kann auftreten, wenn ein Dienst (der Anruf-Dienst) einen anderen Dienst anruft (den aufgerufenen Dienst). Der Anruf-Dienst kann so manipuliert werden, dass er seine Berechtigungen verwendet, um auf die Ressourcen eines anderen Kunden zu reagieren, auf die er sonst nicht zugreifen dürfte. Um dies zu verhindern, AWS bietet Tools, mit denen Sie Ihre Daten für alle Dienste mit Dienstprinzipalen schützen können, denen Zugriff auf Ressourcen in Ihrem Konto gewährt wurde.

Wir empfehlen, die Kontextschlüssel `aws:SourceArn`und die `aws:SourceAccount`globalen Bedingungsschlüssel in Ressourcenrichtlinien zu verwenden, um die Berechtigungen einzuschränken, AWS Proton die der Ressource einen anderen Dienst gewähren. Wenn der `aws:SourceArn`-Wert die Konto-ID nicht enthält, z. B. einen Amazon-S3-Bucket-ARN, müssen Sie beide globale Bedingungskontextschlüssel verwenden, um Berechtigungen einzuschränken. Wenn Sie beide globale Bedingungskontextschlüssel verwenden und der `aws:SourceArn`-Wert die Konto-ID enthält, müssen der `aws:SourceAccount`-Wert und das Konto im `aws:SourceArn`-Wert dieselbe Konto-ID verwenden, wenn sie in der gleichen Richtlinienanweisung verwendet wird. Verwenden Sie `aws:SourceArn`, wenn Sie nur eine Ressource mit dem betriebsübergreifenden Zugriff verknüpfen möchten. Verwenden Sie `aws:SourceAccount`, wenn Sie zulassen möchten, dass Ressourcen in diesem Konto mit der betriebsübergreifenden Verwendung verknüpft werden.

Der Wert von `aws:SourceArn` muss eine Ressource sein, die AWS Proton speichert.

Der effektivste Weg, um sich vor dem Confused-Deputy-Problem zu schützen, ist die Verwendung des globalen Bedingungskontext-Schlüssels `aws:SourceArn` mit dem vollständigen ARN der Ressource. Wenn Sie den vollständigen ARN der Ressource nicht kennen oder wenn Sie mehrere Ressourcen angeben, verwenden Sie den globalen Bedingungskontext-Schlüssel `aws:SourceArn` mit Platzhaltern (`*`) für die unbekannten Teile des ARN. Beispiel, `arn:aws::proton:*:`*`123456789012`*`:environment/*`.

Das folgende Beispiel zeigt, wie Sie die Kontextschlüssel `aws:SourceArn` und die `aws:SourceAccount` globale Bedingung verwenden können, AWS Proton um das Problem des verwirrten Stellvertreters zu vermeiden.

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 Bereitstellung von benutzerdefiniertem Amazon VPC-Support

AWS Proton CodeBuild Beim Provisioning werden beliebige vom Kunden bereitgestellte CLI-Befehle in einem CodeBuild Projekt ausgeführt, das sich im Environment-Konto befindet. AWS Proton Diese Befehle verwalten Ressourcen in der Regel mithilfe eines IaC-Tools (Infrastructure as Code) wie CDK. Wenn Sie über Ressourcen in einer Amazon VPC verfügen, CodeBuild können Sie möglicherweise nicht darauf zugreifen. Um dies zu ermöglichen, CodeBuild unterstützt es die Ausführung innerhalb einer bestimmten Amazon VPC. Einige Beispiele für Anwendungsfälle sind:

- Rufen Sie Abhängigkeiten aus selbst gehosteten, internen Artefakt-Repositorys ab, z. B. `PyPI` für Python, `Maven` für Java und für Node.js `npm`
- CodeBuild muss auf einen Jenkins-Server in einer bestimmten Amazon VPC zugreifen, um eine Pipeline zu registrieren.

- Greifen Sie auf Objekte in einem Amazon S3 S3-Bucket zu, der so konfiguriert ist, dass der Zugriff nur über einen Amazon VPC-Endpunkt möglich ist.
- Führen Sie Integrationstests von Ihrem Build aus anhand von Daten in einer Amazon RDS-Datenbank durch, die in einem privaten Subnetz isoliert ist.

Weitere Informationen finden Sie CodeBuild in der VPC-Dokumentation.

Wenn Sie möchten, dass CodeBuild Provisioning in einer benutzerdefinierten VPC ausgeführt wird, AWS Proton bietet dies eine einfache Lösung. Zunächst müssen Sie die VPC-ID, Subnetze und Sicherheitsgruppen zur Umgebungsvorlage hinzufügen. Als Nächstes geben Sie diese Werte in die Umgebungsspezifikation ein. Dies führt dazu, dass für Sie ein CodeBuild Projekt erstellt wird, das auf eine bestimmte VPC abzielt.

Aktualisierung der Umgebungsvorlage

Schema

Die VPC-ID, die Subnetze und die Sicherheitsgruppen müssen dem Vorlagenschema hinzugefügt werden, damit sie in der Umgebungsspezifikation existieren können.

Ein Beispiel: `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
```

Dadurch werden drei neue Eigenschaften hinzugefügt, die vom Manifest verwendet werden:

- `codebuild_vpc_id`
- `codebuild_subnets`
- `codebuild_security_groups`

Manifest

Um die Amazon VPC-Einstellungen zu konfigurieren CodeBuild, `project_properties` steht im Vorlagenmanifest eine optionale Eigenschaft namens zur Verfügung. Der Inhalt von `project_properties` wird dem CloudFormation Stack hinzugefügt, der das CodeBuild Projekt erstellt. Dadurch ist es möglich, nicht nur Amazon CloudFormation VPC-Eigenschaften hinzuzufügen, sondern auch alle unterstützten CodeBuild CloudFormation Eigenschaften, wie z. B. Build-Timeout. Dieselben Daten, die für `proton-inputs.json` bereitgestellt wurden, werden für die Werte von zur Verfügung gestellt. `project_properties`

Fügen Sie diesen Abschnitt zu Ihrem hinzu`manifest.yaml`:

```
project_properties:
          VpcConfig:
            VpcId: "{{ environment.inputs.codebuild_vpc_id }}"
            Subnets: "{{ environment.inputs.codebuild_subnets }}"
            SecurityGroupIds: "{{ environment.inputs.codebuild_security_groups }}"
```

Das Ergebnis `manifest.yaml` könnte wie folgt aussehen:

```
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 }}"
```

Die Umgebung erstellen

Wenn Sie eine Umgebung mit Ihrer CodeBuild Provisioning VPC-fähigen Vorlage erstellen, müssen Sie die Amazon VPC-ID, Subnetze und Sicherheitsgruppen angeben.

Führen Sie den folgenden Befehl aus, um eine Liste aller Amazon VPC IDs in Ihrer Region abzurufen:

```
aws ec2 describe-vpcs
```

Um eine Liste aller Subnetze zu erhalten IDs, führen Sie folgenden Befehl aus:

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=vpc-id"
```

> **Important**
>
> Schließt nur private Subnetze ein. CodeBuild schlägt fehl, wenn Sie öffentliche Subnetze bereitstellen. Öffentliche Subnetze haben eine Standardroute zu einem Internet Gateway, private Subnetze dagegen nicht.

Führen Sie den folgenden Befehl aus, um die Sicherheitsgruppe abzurufen. IDs Diese IDs können auch abgerufen werden über AWS-Managementkonsole:

```
aws ec2 describe-security-groups --filters "Name=vpc-id,Values=vpc-id"
```

Die Werte werden wie folgt aussehen:

```
vpc-id: vpc-045ch35y28dec3a05
subnets:
  - subnet-04029a82e6ae46968
```

```
  - subnet-0f500a9294fc5f26a
 security-groups:
  - sg-03bc4c4ce32d67e8d
```

Sicherstellung von CodeBuild Genehmigungen

Für die Unterstützung von Amazon VPC sind bestimmte Berechtigungen erforderlich, z. B. die Möglichkeit, eine Elastic Network Interface zu erstellen.

Wenn die Umgebung in der Konsole erstellt wird, geben Sie diese Werte während des Assistenten zur Umgebungserstellung ein. Wenn Sie die Umgebung programmgesteuert erstellen möchten, `spec.yaml` sieht sie wie folgt aus:

```
proton: EnvironmentSpec

spec:
  codebuild_vpc_id: vpc-045ch35y28dec3a05
  codebuild_subnets:
    - subnet-04029a82e6ae46968
    - subnet-0f500a9294fc5f26a
  codebuild_security_groups:
    - sg-03bc4c4ce32d67e8d
```

AWS Proton Ressourcen und Tagging

AWS Proton Zu den Ressourcen, denen ein Amazon Resource Name (ARN) zugewiesen wurde, gehören Umgebungsvorlagen und ihre Haupt- und Nebenversionen, Service-Vorlagen und ihre Haupt- und Nebenversionen, Umgebungen, Services, Service-Instances, Komponenten und Repositorys. Sie können diese Ressourcen taggen, um sie besser organisieren und identifizieren zu können. Mithilfe von Tags können Sie Ressourcen nach Zweck, Eigentümer, Umgebung oder anderen Kriterien kategorisieren. Weitere Informationen finden Sie unter Tagging-Strategien in . Um Ihre AWS Proton Ressourcen nachzuverfolgen und zu verwalten, können Sie die in den folgenden Abschnitten beschriebenen Tagging-Funktionen verwenden.

AWS Tagging

Sie können Ihren AWS Ressourcen Metadaten in Form von Tags zuweisen. Jedes Tag besteht aus einem vom Kunden definierten Schlüssel und einem optionalen Wert. Mit Tags können Sie Ressourcen verwalten, identifizieren, organisieren, suchen und filtern.

> **Important**
>
> Fügen Sie keine personenbezogenen Daten (Personally Identifiable Information, PII) oder andere vertrauliche Informationen in Tags hinzu. Tags sind für viele AWS-Services zugänglich, einschließlich der Abrechnung. Tags sind nicht dafür vorgesehen, für private oder sensible Daten verwendet zu werden.

Jedes -Tag besteht aus zwei Teilen.

- einem Tag-Schlüssel (z. B. `CostCenter`, `Environment` oder `Project`). Bei Tag-Schlüsseln wird zwischen Groß- und Kleinschreibung unterschieden.
- Ein Tag-Wert (optional) (z. B. `111122223333` oder`Production`). Wie bei Tag-Schlüsseln wird auch bei Tag-Werten zwischen Groß- und Kleinschreibung unterschieden.

Die folgenden grundlegenden Anforderungen an die Benennung und Verwendung gelten für Tags.

- Eine Ressource kann bis zu 50 Tags besitzen, die von Benutzern erstellt wurden.

> **Note**
>
> Vom System erstellte Tags, die mit dem `aws:` Präfix beginnen, sind für die AWS Verwendung reserviert und werden nicht auf dieses Limit angerechnet. Tags, die mit dem einem `aws:`-Präfix beginnen, können nicht bearbeitet oder gelöscht werden.

- Jeder Tag (Markierung) muss für jede Ressource eindeutig sein. Jeder Tag (Markierung) kann nur einen Wert haben.
- Der Tag-Schlüssel muss mindestens 1 und maximal 128 Unicode-Zeichen in UTF-8 enthalten.
- Der Tag-Wert muss mindestens 1 und maximal 256 Unicode-Zeichen in UTF-8 enthalten.
- Zulässige Zeichen in Tags sind Buchstaben, Zahlen, in UTF-8 darstellbare Leerzeichen und die folgenden Zeichen: * _.:/= + - @.

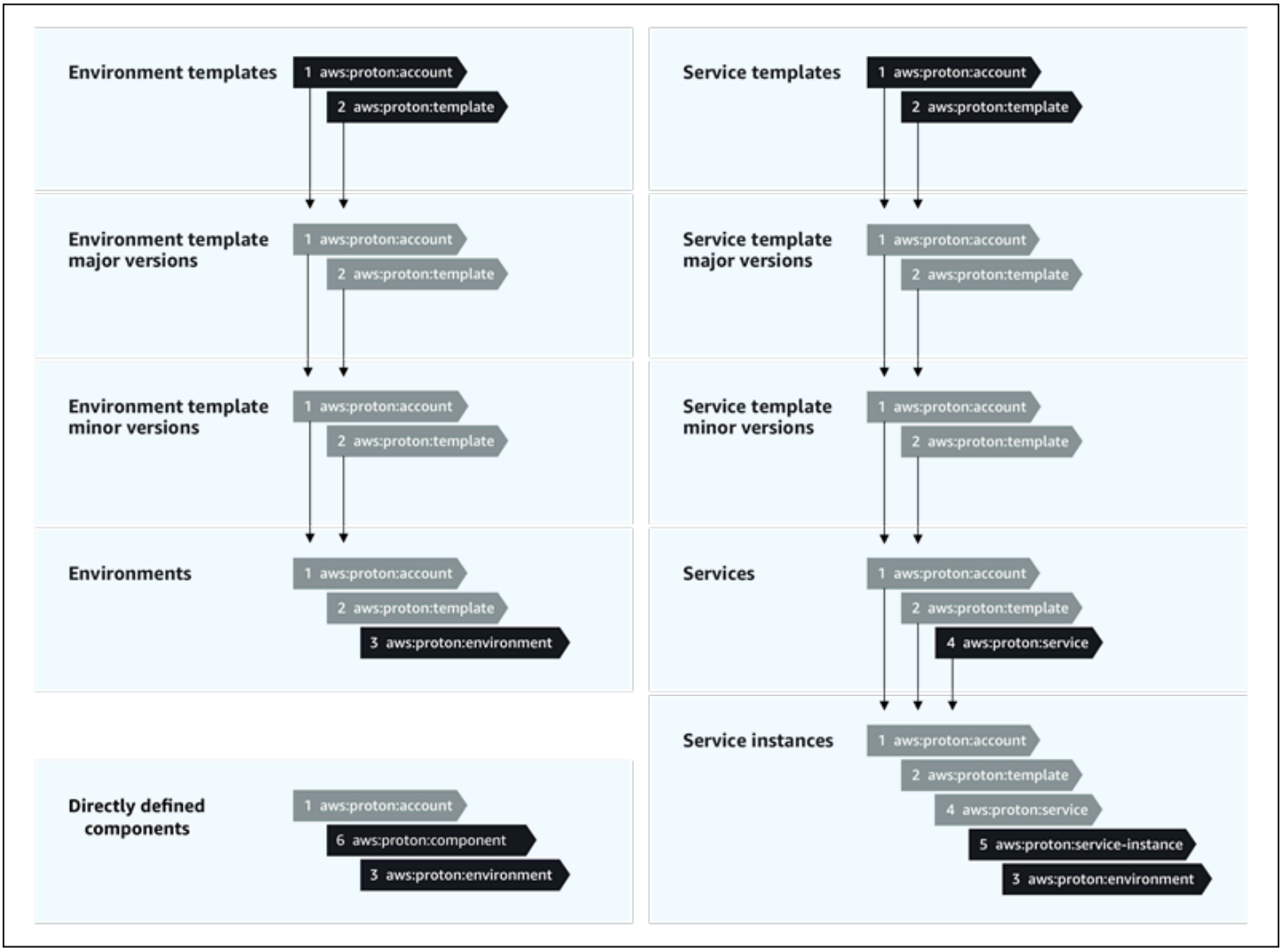
AWS Proton Taggen

Mit AWS Proton können Sie sowohl die von Ihnen erstellten Tags als auch die Tags verwenden, die AWS Proton automatisch für Sie generiert werden.

AWS Proton AWS verwaltete Tags

Wenn Sie eine AWS Proton Ressource erstellen, AWS Proton werden automatisch AWS verwaltete Tags für Ihre neue Ressource generiert, wie in der folgenden Abbildung dargestellt. AWS Verwaltete Tags werden später auf andere AWS Proton Ressourcen übertragen, die auf Ihrer neuen Ressource basieren. Beispielsweise werden verwaltete Tags aus einer Umgebungsvorlage auf ihre Versionen übertragen, und verwaltete Tags aus einem Dienst werden auf seine Dienstinstanzen übertragen.

> **Note**
>
> AWS verwaltete Tags werden nicht für Verbindungen mit Umgebungskonten generiert. Weitere Informationen finden Sie unter the section called “Kontoverbindungen”.



Weitergabe von Tags an bereitgestellte Ressourcen

Wenn bereitgestellte Ressourcen, wie sie in Service- und Umgebungsvorlagen definiert sind, AWS Tagging unterstützen, werden die verwalteten Tags als vom Kunden AWS verwaltete Tags auf die bereitgestellten Ressourcen übertragen. Diese Tags werden nicht an eine bereitgestellte Ressource weitergegeben, die Tagging nicht unterstützt. AWS

AWS Proton wendet Tags auf Ihre Ressourcen nach AWS Proton Konten, registrierten Vorlagen und bereitgestellten Umgebungen sowie nach Diensten und Dienstinstanzen an, wie in der folgenden Tabelle beschrieben. Sie können AWS verwaltete Tags verwenden, um Ihre AWS Proton Ressourcen anzuzeigen und zu verwalten, aber Sie können sie nicht ändern.

AWS Schlüssel für verwaltete Tags	Vom Kunden verwalteter Schlüssel weitergegeben	Description
`aws:proton:account`	`proton:account`	Das AWS Konto, das Ressourcen erstellt und bereitstellt AWS Proton .
`aws:proton:template`	`proton:template`	Der ARN einer ausgewählten Vorlage.
`aws:proton:environment`	`proton:environment`	Der ARN einer ausgewählten Umgebung.
`aws:proton:service`	`proton:service`	Der ARN eines ausgewählten Dienstes.
`aws:proton:service-instance`	`proton:service-instance`	Der ARN einer ausgewählten Dienstinstanz.
`aws:proton:component`	`proton:component`	Der ARN einer ausgewählten Komponente.

Im Folgenden finden Sie ein Beispiel für ein AWS verwaltetes Tag für eine AWS Proton Ressource.

```
"aws:proton:template" = "arn:aws:proton:region-id:account-id:environment-template/env-template"
```

Im Folgenden finden Sie ein Beispiel für ein vom Kunden verwaltetes Tag, das auf eine bereitgestellte Ressource angewendet wurde und über ein AWS verwaltetes Tag weitergegeben wurde.

```
"proton:environment:database" = "arn:aws:proton:region-id:account-id:rds/env-db"
```

AWS Proton Wendet bei AWS-managed provisioning weitergegebene Tags direkt auf bereitgestellte Ressourcen an.

Stellt bei selbstverwalteter Bereitstellung propagierte Tags zusammen mit AWS Proton den gerenderten IaC-Dateien zur Verfügung, die es im Provisioning Pull Request (PR) übermittelt. Tags werden in der String-Map-Variablen bereitgestellt. `proton_tags` Wir empfehlen, dass Sie in

Ihrer Terraform-Konfiguration auf diese Variable verweisen, um AWS Proton Tags einzubeziehen. `default_tags` Dadurch werden AWS Proton Tags an alle bereitgestellten Ressourcen weitergegeben.

Das folgende Beispiel zeigt diese Methode der Tag-Weitergabe in einer Terraform-Umgebungsvorlage.

Hier ist die `proton_tags` Variablendefinition:

proton.environment.variables.tf:

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

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

So werden Tag-Werte dieser Variablen zugewiesen:

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"
  }
}
```

Und so können Sie Ihrer Terraform-Konfiguration AWS Proton Tags hinzufügen, sodass sie zu den bereitgestellten Ressourcen hinzugefügt werden:

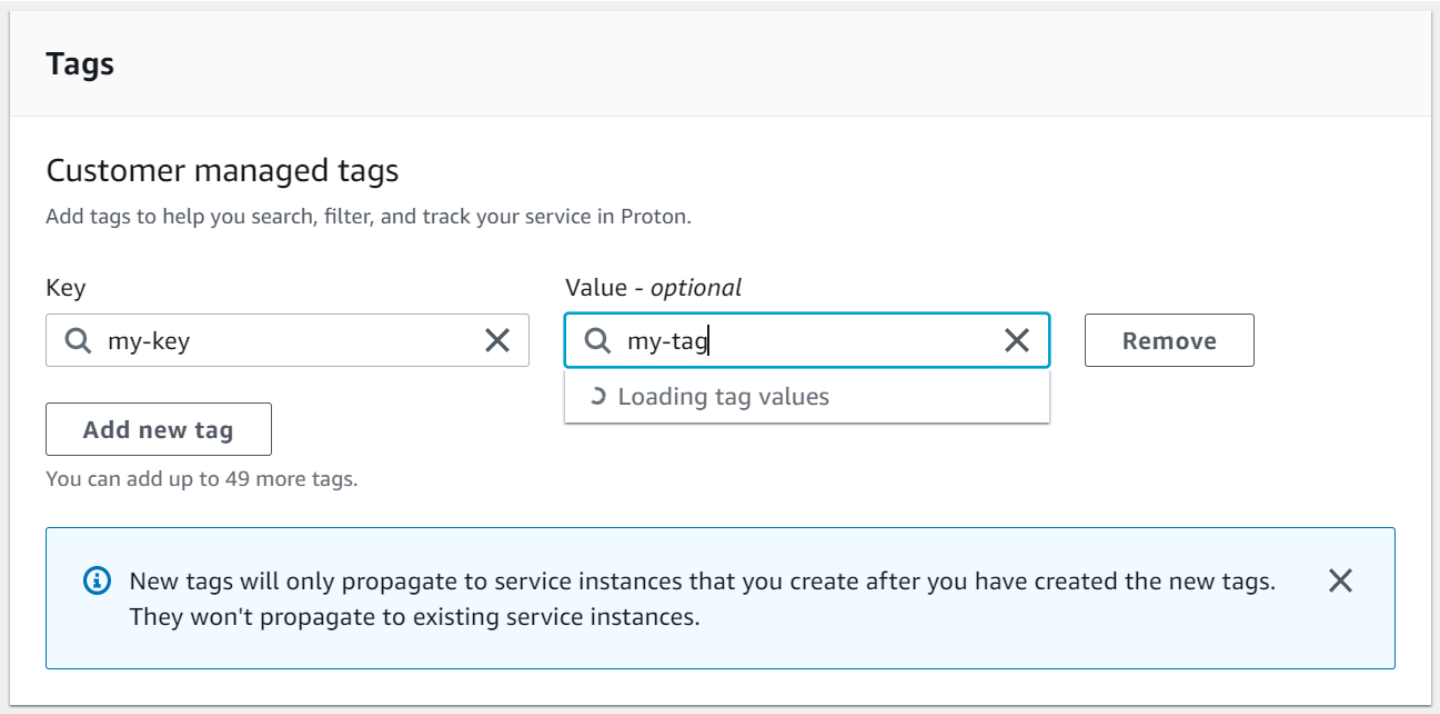
```
# Configure the AWS Provider
provider "aws" {
  region = var.aws_region
  default_tags {
    tags = var.proton_tags
  }
}
```

Vom Kunden verwaltete Tags

Jede AWS Proton Ressource hat ein maximales Kontingent von 50 vom Kunden verwalteten Tags. Vom Kunden verwaltete Tags werden auf die gleiche Weise an untergeordnete AWS Proton Ressourcen weitergegeben wie AWS verwaltete Tags, außer dass sie nicht auf vorhandene AWS Proton Ressourcen oder bereitgestellte Ressourcen übertragen werden. Wenn Sie ein neues Tag auf eine AWS Proton Ressource mit vorhandenen untergeordneten Ressourcen anwenden und möchten, dass die vorhandenen untergeordneten Ressourcen mit dem neuen Tag gekennzeichnet werden, müssen Sie jede vorhandene untergeordnete Ressource manuell kennzeichnen, indem Sie die Konsole oder verwenden. AWS CLI

Erstellen Sie Tags mit der Konsole und der CLI

Wenn Sie eine AWS Proton Ressource mithilfe der Konsole erstellen, haben Sie die Möglichkeit, vom Kunden verwaltete Tags entweder auf der ersten oder zweiten Seite des Erstellungsvorgangs zu erstellen, wie im folgenden Konsolen-Snapshot gezeigt. Wählen Sie Neues Tag hinzufügen, geben Sie den Schlüssel und den Wert ein und fahren Sie fort.



Nachdem Sie mit der AWS Proton Konsole eine neue Ressource erstellt haben, können Sie die Liste der AWS verwalteten und kundenverwalteten Tags auf der Detailseite einsehen.

Erstellen oder bearbeiten Sie ein Tag

1. Öffnen Sie in der AWS Proton Konsole eine AWS Proton Ressourcendetailseite, auf der Sie eine Liste von Tags sehen können.
2. Wählen Sie Tags verwalten aus.
3. Auf der Seite „Tags verwalten“ können Sie Tags anzeigen, erstellen, entfernen und bearbeiten. Sie können die oben aufgeführten AWS verwalteten Tags nicht ändern. Sie können jedoch die vom Kunden verwalteten Tags mit Bearbeitungsfeldern ergänzen und ändern, die hinter den AWS verwalteten Stichwörtern aufgeführt sind.

   Wählen Sie Neues Tag hinzufügen, um ein neues Tag zu erstellen.
4. Geben Sie einen Schlüssel und einen Wert für das neue Tag ein.
5. Um ein Tag zu bearbeiten, geben Sie einen Wert in das Feld Tag-Wert für einen ausgewählten Schlüssel ein.
6. Um ein Tag zu löschen, wählen Sie Entfernen für ein ausgewähltes Tag.
7. Wenn Sie Ihre Änderungen abgeschlossen haben, wählen Sie Änderungen speichern.

Erstellen Sie Tags mit dem AWS Proton AWS CLI

Sie können Tags mit dem anzeigen, erstellen, entfernen und bearbeiten AWS Proton AWS CLI.

Sie können ein Tag für eine Ressource erstellen oder bearbeiten, wie im folgenden Beispiel gezeigt.

```
$ 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"}]'
```

Sie können ein Tag für eine Ressource entfernen, wie im nächsten Beispiel gezeigt.

```
$ aws proton untag-resource \
    --resource-arn "arn:aws:proton:region-id:account-id:service-template/webservice" \
    --tag-keys '["mykey1","mykey2"]'
```

Sie können Tags für eine Ressource auflisten, wie im letzten Beispiel gezeigt.

```
$ aws proton list-tags-for-resource \
    --resource-arn "arn:aws:proton:region-id:account-id:service-template/webservice"
```

Problembehebung AWS Proton

Erfahren Sie, wie Sie Probleme mit beheben können AWS Proton.

Themen

Bereitstellungsfehler, die auf CloudFormation dynamische Parameter verweisen

Wenn Sie Bereitstellungsfehler sehen, die auf Ihre CloudFormation dynamischen Variablen verweisen, stellen Sie sicher, dass es sich um Jinja-Escapes handelt. Diese Fehler können durch eine Fehlinterpretation Ihrer dynamischen Variablen durch Jinja verursacht werden. Die CloudFormation dynamische Parametersyntax ist der Jinja-Syntax, die Sie mit Ihren Parametern verwenden, sehr ähnlich. AWS Proton

Beispiel für CloudFormation eine dynamische Variablensyntax:

`'{{resolve:secretsmanager:MySecret:SecretString:password:EXAMPLE1-90ab-cdef-fedc-ba987EXAMPLE}}'`.

Beispiel für eine Jinja-Syntax für einen AWS Proton Parameter:

`'{{ service_instance.environment.outputs.env-outputs }}'`.

Um diese Fehlinterpretationsfehler zu vermeiden, maskiert Jinja Ihre CloudFormation dynamischen Parameter, wie in den folgenden Beispielen gezeigt.

Dieses Beispiel stammt aus dem CloudFormation Benutzerhandbuch. Die Segmente AWS Secrets Manager Secret-Name und JSON-Key können verwendet werden, um die im Secret gespeicherten Anmeldedaten abzurufen.

```
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}}'
```

Um die CloudFormation dynamischen Parameter zu umgehen, können Sie zwei verschiedene Methoden verwenden:

- Schließe einen Block ein zwischen`{% 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 %}'
```

- Schließen Sie einen Parameter ein zwischen: `"{{ }}"`

```
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}}' }}"
```

Weitere Informationen finden Sie unter Jinja escaping.

AWS Proton Kontingente

In der folgenden Tabelle sind die AWS Proton Kontingente aufgeführt. Alle Werte verstehen sich pro AWS Konto und pro unterstützter AWS Region.

Ressourcenkontingent	Standardlimit	Anpassbar?
Maximale Größe des Vorlagenpakets	10 MB	✗ Nein
Maximale Größe der Vorlagenmanifestdatei	2 MB	✗ Nein
Maximale Größe der Vorlagenschemadatei	2 MB	✗ Nein
Maximale Größe jeder Vorlagendatei	2 MB	✗ Nein
Maximale Länge jedes Vorlagennamens	100 Zeichen	✗ Nein
Maximale Anzahl von CloudFormation Vorlagendateien pro Paket	1	✗ Nein
Maximale Anzahl registrierter Vorlagen pro Konto-, Service- und Umgebungsvorlagen zusammen	1000	✓ Ja
Maximale Anzahl registrierter Vorlageversionen pro Vorlage	1000	✓ Ja
Maximale Anzahl von Dateien pro CodeBuild Provisioning-Paket	500	✗ Nein
Die maximale Anzahl an Umgebungen pro Konto	1000	✓ Ja
Maximale Anzahl der Services pro Konto	1000	✓ Ja
Maximale Anzahl von Service-Instances pro Service	20	✓ Ja
Maximale Anzahl der Komponenten pro Konto	1000	✓ Ja
Maximale Anzahl von Umgebungskontoverbindungen pro Umgebungskonto	1000	✓ Ja

Dokumentverlauf

In der folgenden Tabelle werden die wichtigen Änderungen an der Dokumentation im Zusammenhang mit der neuesten Version von AWS Proton sowie Kundenfeedback beschrieben. Um Benachrichtigungen über Aktualisierungen dieser Dokumentation zu erhalten, können Sie einen RSS-Feed abonnieren.

- API-Version: 2020-07-20

Änderung	Beschreibung	Datum
Mitteilung zum Ende des Supports	Am 7. Oktober 2026 AWS wird der Support für eingestellt. AWS Proton	7. Oktober 2025
Aktualisierung der verwalteten Richtlinien	`AWSProtonCodeBuildProvisioningServiceRolePolicy` Die Richtlinie wurde aktualisiert.	15. Juni 2024
Aktualisierung der verwalteten Richtlinien	`AWSProtonDeveloperAccess` Die Richtlinie wurde aktualisiert.	25. April 2024
Aktualisierung der verwalteten Richtlinien	`AWSProtonFullAccess` Die Richtlinie wurde aktualisiert.	25. April 2024
Aktualisierung der verwalteten Richtlinien	`AWSProtonSyncServiceRolePolicy` Die Richtlinie wurde aktualisiert.	25. April 2024
Aktualisierung der verwalteten Richtlinien	`AWSProtonCodeBuildProvisioningServiceRolePolicy` Die Richtlinie wurde aktualisiert.	12. Mai 2023

Konfigurationen für die Synchronisierung von Diensten.	AWS Proton fügt Unterstützung für Service Sync-Konfigurationen hinzu.	31. März 2023
CodeBuild	AWS Proton fügt Unterstützung für die CodeBuild Bereitstellung hinzu.	16. November 2022
Aktualisierung der verwalteten Richtlinien	Es wurde eine `AWSProtonCodeBuildProvisioningBasicAccess` Richtlinie hinzugefügt, CodeBuild die die Berechtigungen erteilt, die für die Ausführung eines Builds für die AWS Proton CodeBuild Bereitstellung erforderlich sind.	11. November 2022
Verbreitung von Terraform-Tags	Die Übertragung von Terraform-Tags wurde dem Kapitel Tagging hinzugefügt.	16. September 2022
Leitfaden zur API-Migration	Der Leitfaden zur API-Migration vor der GA-Version wurde entfernt.	12. August 2022
AWS Proton Objekte	Es wurde ein Thema über AWS Proton Objekte und ihre Beziehung zu anderen Objekten AWS und Objekten von Drittanbietern hinzugefügt. Siehe AWS Proton Objekte.	29. Juli 2022
Erläuterungen zum verlinkten Repository	Der Zweck verlinkter (registrierter) Repositorien und ihre Verwendung im gesamten Leitfaden wurden klargestellt.	18. Juli 2022

Zusammenführung von Anleitungen	Die beiden separaten Administrator- und Benutzerh andbücher wurden zu einem einzigen Handbuch, dem AWS Proton Benutzerhandbuch, zusammengeführt.	30. Juni 2022
Aktualisierung der verwalteten Richtlinien	Die verwalteten Richtlinien wurden aktualisiert, um Zugriff auf neue AWS Proton API-Operationen zu gewähren und Berechtigungsprobleme bei einigen AWS Proton Konsolenoperationen zu beheben. Weitere Informati onen finden Sie unter AWS Verwaltete Richtlinien für AWS Proton.	20. Juni 2022
Erste Schritte mit der CLI	Aktualisiert Erste Schritte AWS CLI mit einem neuen Tutorial, das die neue Vorlagenb ibliothek verwendet.	14. Juni 2022
Direkt definierte Komponenten	Das Kapitel Komponent en wurde hinzugefügt und entsprechende Änderunge n wurden im gesamten Handbuch vorgenommen.	1. Juni 2022
AWS Proton Vorlagenb ibliothek	Das Thema „ AWS Proton Vorlagenbibliothek“ wurde hinzugefügt.	6. Mai 2022

Allgemeine Verfügbarkeit von Terraform (GA)	Die Bereitstellung von Pull-Requests wurde in selbstverwaltete Bereitstellung umbenannt. Das Thema Bereitstellungsmethoden wurde hinzugefügt.	23. März 2022
Repository-Tagging	Unterstützung für das Taggen von Repository-Ressourcen wurde hinzugefügt. Weitere Informationen findest du unter Einen Link zu deinem Repository erstellen.	23. März 2022
Aktualisierung der Dokumentation	Verbindungskennzeichnung für Umgebungskonten hinzugefügt.	26. November 2021
Vorlagensynchronisierung und Terraform-Vorschau	Automatisierte Vorlagenversionierung mit der Funktion zur Vorlagensynchronisierung für allgemeine Verfügbarkeit und Bereitstellung von Pull-Requests mit Terraform in der Vorschauversion hinzugefügt. Der Leitfaden zur API-Migration ist wieder da.	24. November 2021
Aktualisierungen der Dokumentation	Der Abschnitt EventBridgeTutorial, der Arbeitsablauf „Erste Schritte“, die AWS Proton Funktionsweise und der Abschnitt „Vorlagenpaket“ wurden hinzugefügt.	17. September 2021

AWS Proton Veröffentlichung der Hilfebereiche für Konsolen	Hilfetafeln wurden der Konsole hinzugefügt. Beim Löschen der Konsolenvorlagenversion werden ältere Versionen nicht mehr gelöscht. Der API-Migrationsleitfaden wurde entfernt.	8. September 2021
AWS Proton Version für allgemeine Verfügbarkeit (GA)	Es wurden kontenübergreifende Umgebungen, EventBridge Überwachung, IAM-Bedingungsschlüssel, Unterstützung für Idempotenz und höhere Kontingente hinzugefügt.	9. Juni 2021
Fügen Sie Dienstinstanzen für einen Service hinzu und löschen Sie sie und verwenden Sie die vorhandene externe Infrastruktur für Umgebungen mit AWS Proton	Diese öffentliche Vorschauversion enthält Updates, die es Ihnen ermöglichen, Dienstinstanzen zu einem Service hinzuzufügen und zu löschen, Ihre bestehende externe Infrastruktur in einer AWS Proton Umgebung zu verwenden und Umgebungs-, Dienstinstanz- und Pipeline-Bereitstellungen abzubrechen. AWS Proton unterstützt PrivateLinkjetzt. Eine zusätzliche Löschvalidierung wurde hinzugefügt, um zu verhindern, dass eine Nebenversion versehentlich gelöscht wird, während eine Ressource sie verwendet.	27. April 2021

Taggen mit AWS Proton	Die öffentliche Vorschauversion 2 beinhaltet AWS Proton Tagging und die Möglichkeit, Dienste ohne Service-Pipeline zu starten.	5. März 2021
Erstversion	Die öffentliche Vorschauversion ist jetzt in ausgewählten Regionen verfügbar.	1. Dezember 2020

AWS Glossar

Die neueste AWS Terminologie finden Sie im AWS Glossar in der AWS-Glossar Referenz.