



Guide de l'utilisateur

AWS Proton



AWS Proton: Guide de l'utilisateur

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Les marques et la présentation commerciale d'Amazon ne peuvent être utilisées en relation avec un produit ou un service qui n'est pas d'Amazon, d'une manière susceptible de créer une confusion parmi les clients, ou d'une manière qui dénigre ou discrédite Amazon. Toutes les autres marques commerciales qui ne sont pas la propriété d'Amazon appartiennent à leurs propriétaires respectifs, qui peuvent ou non être affiliés ou connectés à Amazon, ou sponsorisés par Amazon.

Table of Contents

Qu'est-ce que AWS Proton ?	1
Équipes de plateforme	1
Développeurs	2
Flux de travail	2
Configuration	4
Configuration avec IAM	4
Inscrivez-vous à AWS	5
Créer un utilisateur IAM	5
Rôles de service	7
Configuration avec AWS Proton	8
Configuration d'un compartiment Amazon S3	8
Configuration d'une AWS CodeStar connexion	8
Configuration des paramètres du pipeline CI/CD du compte	9
Configuration du AWS CLI	11
Démarrer	12
Prérequis	12
Flux de travail de démarrage	13
Mise en route avec la console	14
Étape 1 : ouvrir la AWS Proton console	15
Étape 2 : Préparez-vous à utiliser les modèles d'exemple	15
Étape 3 : Création d'un modèle d'environnement	15
Étape 4 : Création d'un modèle de service	16
Étape 5 : Création d'un environnement	17
Étape 6 : Facultatif - Création d'un service et déploiement d'une application	18
Étape 7 : Nettoyez.	20
Commencer à utiliser la CLI	21
1. Enregistrer un modèle d'environnement	21
2. Enregistrer un modèle de service	23
3. Déployer un environnement	24
4. Déployer un service	25
5. Nettoyage	27
Bibliothèque de modèles	28
Fonctionnement d'AWS Proton	29
Objets	30

Méthodes de provisionnement	33
AWS-provisionnement géré	35
CodeBuildapprovisionnement	37
Approvisionnement	40
Terminologie AWS Proton	43
Création de modèles et d'ensembles	46
Packs de modèles	46
Paramètres	48
Types de paramètres	49
Utilisation de paramètres	49
Paramètres CloudFormation IaC de l'environnement	53
Paramètres du service CloudFormation IaC	59
Paramètres du composant CloudFormation IaC	61
CloudFormation filtres de paramètres	65
CodeBuild paramètres de provisionnement	72
Paramètres Terraform IaC	74
Infrastructure sous forme de fichiers de code	75
AWS CloudFormation fichiers IaC	76
CodeBuild offre groupée	129
Fichiers Terraform iAC	135
Fichier de schéma	143
Exigences relatives au schéma d'environnement	143
Exigences relatives au schéma de service	147
Manifeste et résumé	150
Résumé du bundle de modèles d'environnement	153
Résumé de l'offre groupée de modèles de services	154
Considérations relatives aux ensembles de modèles	155
Modèles	156
Versions	157
Publier	159
Publier des modèles d'environnement	159
Publier des modèles de services	167
Afficher les modèles	176
Met à jour un modèle	180
Suppression de modèles	182
Configurations de synchronisation de modèles	186

Encourager un commit	186
Modèles de service de synchronisation	187
Considérations concernant la synchronisation des modèles	187
Création	188
Vue	195
Modifier	197
Delete	198
Configurations de synchronisation des services	199
AWS ProtonFichier OPS	199
Création	203
Vue	205
Modifier	206
Delete	207
Environnements	209
Rôles IAM	209
Rôle de service AWS Proton	209
Création	210
Créez et approvisionnez sur le même compte	212
Créer dans un compte et approvisionner dans un autre	214
Provisionnement autogéré	219
Vue	223
Mettre à jour	224
Mettre à jour unAWSenvironnement de provisionnement géré	225
Mettre à jour un environnement de provisionnement autogéré	228
Annuler un déploiement d'environnement en cours	232
Delete	235
Connexions aux comptes	236
Création d'un environnement avec des connexions à des comptes d'environnement	239
Gérer les connexions aux comptes d'environnement	240
Géré par le client	246
Utilisation d'environnements gérés par le client	247
CodeBuildcréation d'un rôle de provisionnement	249
Services	253
Création	253
Présentation d'un service ?	254
Modèles de service	254

Créer un service	255
Vue	259
Modifier	261
Modifier la description du service	261
Ajouter ou supprimer des instances de service	263
Delete	270
Affichage des instances	271
Mise à jour d'instance	273
Mise à jour du pipeline	280
Composants	287
Composants et autres ressources	289
Console AWS Proton	290
AWS ProtonAPI etAWS CLI	291
FAQ sur les composants	292
États des composants	293
Fichiers IAC du composant	295
Utilisation de paramètres avec des composants	295
Création de fichiers iAC robustes	295
AWS CloudFormationExemple de composants	297
Étapes de l'administrateur	297
Étapes du développeur	300
Référentiels	303
Création d'un lien vers le référentiel	304
Afficher les données du référentiel lié	306
Suppression d'un lien vers un référentiel	309
Surveillance	311
Automatisez AWS Proton avec EventBridge	311
Types d'événements	311
AWS Proton exemples d'événements	314
EventBridgeTutorial: Envoyer des alertes Amazon Simple Notification Service en cas de modification AWS Proton de l'état du service	316
Prérequis	316
Étape 1 : Créer une rubrique Amazon SNS et s'y abonner	316
Étape 2 : Enregistrer une règle d'événement	317
Étape 3 : Testez la règle de votre événement	318
Étape 4 : Nettoyer	319

AWS Proton tableau de bord	321
AWS Proton console	321
Sécurité	323
Identity and Access Management (Gestion des identités et des accès)	324
Public ciblé	324
Authentification par des identités	325
Gestion des accès à l'aide de politiques	329
Fonctionnement de AWS Proton avec IAM	332
Exemples de politiques	340
Politiques gérées par AWS	354
Utilisation des rôles liés aux services	369
Résolution des problèmes	377
Analyse de la configuration et des vulnérabilités	379
Protection des données	380
Chiffrement côté serveur au repos	381
Chiffrement en transit	381
AWS Protongestion des clés de chiffrement	381
Contexte de chiffrement AWS Proton	381
Sécurité de l'infrastructure	383
Points de terminaison d'un VPC (AWS PrivateLink)	383
Journalisation et surveillance	386
Résilience	387
AWS Protonsauvegards	387
Bonnes pratiques de sécurité	387
Utilisation du contrôle d'accès IAM	388
N'incorporez pas d'informations d'identification dans vos modèles et vos offres groupées de modèles	388
Utiliser le chiffrement pour protéger les données sensibles	389
UtiliserAWS CloudTrailpour afficher et journaliser les appels API	389
Prévention du député confus entre services	389
Support personnalisé pour Codebuild	390
Mise à jour du modèle d'environnement	391
Identification	395
AWSmarquage	395
AWS Protonmarquage	396
AWS Proton AWSbalises gérées	396

Propagation des balises vers les ressources provisionnées	397
Tags gérées par les clients	400
Création de balises via la console et l'interface de ligne	400
Créez des balises en utilisant le pluginAWS Proton AWS CLI	402
Résolution des problèmes	403
Erreurs de déploiement faisant référence à des paramètresAWS CloudFormation dynamiques	403
Quotas AWS Proton	405
Historique de document	407
Glossaire AWS	412
.....	cdxiii

Qu'est-ce que AWS Proton ?

AWS Proton est :

- Infrastructure automatisée sous forme de provisionnement de code et de déploiement d'applications sans serveur et basées sur des conteneurs

Le service AWS Proton est un cadre d'automatisation à deux volets. En tant qu'administrateur, vous créez des modèles de service versionnés qui définissent une infrastructure normalisée et des outils de déploiement pour les applications sans serveur et basées sur des conteneurs. En tant que développeur d'applications, vous pouvez sélectionner des modèles de service pour automatiser les déploiements de vos applications ou services.

AWS Proton identifie toutes les recettes existantes d'instances de service qui utilisent une version de modèle obsolète pour vous. En tant qu'administrateur, vous pouvez demander à AWS Proton de les mettre à niveau en un clic.

- Infrastructure normalisée

Les équipes de plateforme peuvent utiliser AWS Proton et une infrastructure versionnée sous forme de modèles de code. Ils peuvent utiliser ces modèles pour définir et gérer des piles d'applications standard qui contiennent l'architecture, les ressources d'infrastructure et le pipeline de déploiement logiciel CI/CD.

- Déploiements intégrés à CI/CD

Lorsque les développeurs utilisent l'interface en libre-service AWS Proton pour sélectionner un modèle de service, ils sélectionnent une définition de pile d'applications normalisée pour leurs déploiements de code. AWS Proton provisionne automatiquement les ressources, configure le pipeline CI/CD et déploie le code dans l'infrastructure définie.

AWS Proton pour les équipes de plateforme

En tant qu'administrateur, vous ou les membres de l'équipe de votre plateforme, créez des modèles d'environnement et des modèles de service contenant l'infrastructure sous forme de code. Le modèle d'environnement définit l'infrastructure partagée utilisée par plusieurs applications ou ressources. Le modèle de service définit le type d'infrastructure nécessaire au déploiement et à la maintenance d'une application ou d'un microservice unique dans un environnement. Un service AWS Proton est une instantiation d'un modèle de service, qui comprend normalement plusieurs instances de service et

Un AWS Proton Instance de service est une instantiation d'un Modèle de service dans un environnement. Vous ou les autres membres de votre équipe pouvez spécifier des modèles d'environnements compatibles avec un Modèle de service. Pour plus d'informations sur les modèles, voir [AWS Proton Modèles](#).

Vous pouvez utiliser les infrastructures suivantes en tant que fournisseurs de code avec AWS Proton :

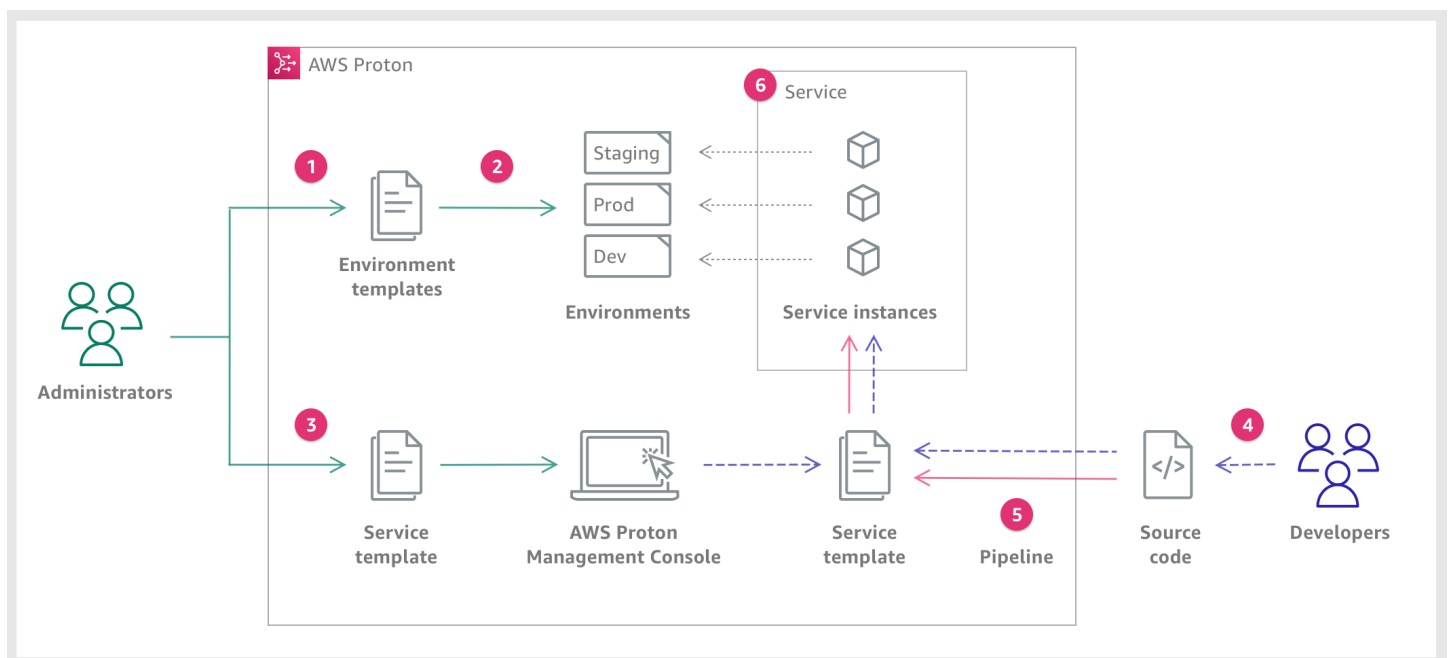
- [AWS CloudFormation](#)
- [Terraform](#)

AWS Proton pour les développeurs

En tant que développeur d'applications, vous sélectionnez un Modèle de service dans AWS Proton pour créer un service qui déploie et gère votre application dans une Instance de service. Un AWS Proton service est une instantiation d'un Modèle de service, qui comprend normalement plusieurs Instances de service et un gazoduc.

AWS Proton flux de travail

Le schéma suivant est une visualisation des principaux concepts AWS Proton abordés dans le paragraphe précédent. Il offre également un aperçu général de ce qui constitue un AWS Proton Flux de travail.



- 1** En tant que **Administrateur**, vous créez et enregistrez un **Modèle d'environnement** avec **AWS Proton**, qui définit les ressources partagées.
- 2** **AWS Proton** déploie une ou plusieurs **Environnements**, sur la base d'un **Modèle d'environnement**.
- 3** En tant que **Administrateur**, vous créez et enregistrez un **Modèle de service** avec **AWS Proton**, qui définit l'infrastructure associée, la surveillance et les ressources **CI/CD** ainsi que les **Modèles d'environnement**.
- 4** En tant que **Developer**, vous sélectionnez un **Modèle de service** et fournissez un lien vers votre **Code source repository**.
- 5** **AWS Proton** dispose de **Service avec Gazoduc CI/CD** pour vos recettes **Instances de service**.
- 6** **AWS Proton** fournit et gère **Service** et l'**Instances de service** qui exécutent le **Code source** tel qu'il a été défini dans le **Modèle de service**. Une **Instance de service** est une instantiation du **Modèle de service** dans un **Environnement** pour une seule étape d'un **Pipeline** (par exemple **Prod**).

Configuration

Effectuez les tâches décrites dans cette section afin de pouvoir créer et enregistrer des modèles de service et d'environnement. Vous en avez besoin pour déployer des environnements et des services AWS Proton.

Note

Nous offrons AWS Proton sans frais supplémentaires. Vous pouvez créer, enregistrer et gérer des modèles de service et d'environnement gratuitement. Vous pouvez également compter sur l'autogestion AWS Proton de ses propres opérations, telles que le stockage, la sécurité et le déploiement. Les seules dépenses que vous encourez lors de l'utilisation AWS Proton sont les suivantes.

- Coûts liés au déploiement et à l'utilisation AWS Cloud des ressources que vous avez AWS Proton chargées de déployer et de gérer pour vous.
- Coûts liés au maintien d'une AWS CodeStar connexion à votre référentiel de code.
- Coûts de maintenance d'un compartiment Amazon S3, si vous utilisez un compartiment pour fournir des entrées à AWS Proton. Vous pouvez éviter ces coûts si vous passez à [the section called “Configurations de synchronisation de modèles”](#) l'utilisation des référentiels Git pour votre [the section called “Packs de modèles”](#).

Rubriques

- [Configuration avec IAM](#)
- [Configuration avec AWS Proton](#)

Configuration avec IAM

Lorsque vous vous inscrivez AWS, vous êtes automatiquement Compte AWS inscrit à tous les services de AWS, y compris AWS Proton. Seuls les services et les ressources que vous utilisez vous sont facturés.

 Note

Vous et votre équipe, y compris les administrateurs et les développeurs, devez tous avoir le même compte.

Inscrivez-vous à AWS

Si vous n'avez pas de compte Compte AWS, procédez comme suit pour en créer un.

Pour s'inscrire à un Compte AWS

1. Ouvrez <https://portal.aws.amazon.com/billing/signup>.
2. Suivez les instructions en ligne.

Dans le cadre de la procédure d'inscription, vous recevrez un appel téléphonique et vous saisirez un code de vérification en utilisant le clavier numérique du téléphone.

Lorsque vous souscrivez à un Compte AWS, un Utilisateur racine d'un compte AWS est créé. Par défaut, seul l'utilisateur root a accès à l'ensemble des Services AWS et des ressources de ce compte. En tant que bonne pratique de sécurité, [attribuer un accès administratif à un utilisateur administratif](#), et utilisez uniquement l'utilisateur root pour effectuer [tâches nécessitant un accès utilisateur root](#).

Créer un utilisateur IAM

Afin de créer un utilisateur administrateur, choisissez l'une des options suivantes :

Choisissez un moyen de gérer votre administrateur	Pour	Bit	Vous pouvez également
<p>Dans IAM Identity Center (Recommandé)</p>	<p>Utiliser des identifiants à court terme pour accéder à AWS.</p> <p>Telles sont les meilleures pratiques en matière de sécurité. Pour plus d'informations sur les bonnes pratiques, consultez Security best practices in IAM (Bonnes pratiques de sécurité dans IAM) dans le Guide de l'utilisateur IAM.</p>	<p>Suivre les instructions de la section Mise en route dans le AWS IAM Identity Center Guide de l'utilisateur.</p>	<p>Configuration de l'accès par programmation en Configurant le AWS CLI à utiliser AWS IAM Identity Center dans le AWS Command Line Interface Guide de l'utilisateur.</p>
<p>Dans IAM (Non recommandé)</p>	<p>Utiliser des identifiants à long terme pour accéder à AWS.</p>	<p>Suivre les instructions relatives à la Création de votre premier groupe utilisateur administrateur et utilisateur IAM dans le Guide de l'utilisateur IAM.</p>	<p>Configuration de l'accès par programmation via la Gestion des clés d'accès pour les utilisateurs IAM dans le Guide de l'utilisateur IAM.</p>

Configuration des rôles AWS Proton de service

Il existe quelques rôles IAM que vous souhaitez peut-être créer pour différentes parties de votre AWS Proton solution. Vous pouvez les créer à l'avance à l'aide de la console IAM, ou vous pouvez utiliser la AWS Proton console pour les créer pour vous.

Créez des rôles d'AWS Proton environnement AWS Proton pour permettre d'effectuer des appels d'API vers d'autres services de calcul et de stockage Services AWS AWS CloudFormation AWS CodeBuild, similaires et divers, en votre nom afin de vous fournir des ressources. [Un rôle de provisionnement AWS géré est requis lorsqu'un environnement ou l'une des instances de service qui y sont exécutées utilise AWS le provisionnement géré.](#) Un CodeBuild rôle est requis lorsqu'un environnement ou l'une de ses instances de service utilise le [CodeBuild provisionnement](#). Pour en savoir plus sur les rôles AWS Proton environnementaux, consultez [the section called “Rôles IAM”](#). Lorsque vous [créez un environnement](#), vous pouvez utiliser la AWS Proton console pour choisir un rôle existant pour l'un de ces deux rôles ou pour créer un rôle doté de privilèges administratifs pour vous.

De même, créez des rôles de AWS Proton pipeline pour AWS Proton permettre d'effectuer des appels d'API vers d'autres services en votre nom afin de vous fournir un pipeline CI/CD. Pour en savoir plus sur les rôles du AWS Proton pipeline, voir [the section called “Rôles des services de pipeline”](#). Pour plus d'informations sur la configuration des paramètres CI/CD, consultez [the section called “Configuration des paramètres du pipeline CI/CD du compte”](#)

Note

Comme nous ne savons pas quelles ressources vous allez définir dans vos AWS Proton modèles, les rôles que vous créez à l'aide de la console disposent d'autorisations étendues et peuvent être utilisés à la fois comme rôles de service de AWS Proton pipeline et de rôles de AWS Proton service. Pour les déploiements de production, nous vous recommandons de limiter les autorisations aux ressources spécifiques qui seront déployées en créant des politiques personnalisées pour les rôles de service de AWS Proton pipeline et les rôles de service d'AWS Proton environnement. Vous pouvez créer et personnaliser ces rôles à l'aide de AWS CLI ou IAM. Pour plus d'informations, consultez [Fonctions de service pour AWS Proton](#) et [Créer un service](#).

Configuration avec AWS Proton

Si vous souhaitez utiliser les AWS Proton API AWS CLI pour exécuter, vérifiez que vous les avez installées. Si vous ne l'avez pas encore installé, consultez [Configuration du AWS CLI](#).

AWS Proton configuration spécifique :

- Pour créer et gérer des modèles :
 - Si vous utilisez des [configurations de synchronisation de modèles](#), configurez une [AWS CodeStar connexion](#).
 - Sinon, configurez un compartiment [Amazon S3](#).
- Pour approvisionner l'infrastructure :
 - [Pour un provisionnement autogéré, vous devez configurer une AWS CodeStar connexion.](#)
- (Facultatif) Pour approvisionner des pipelines :
 - [Pour le provisionnement AWSgéré et le provisionnement CodeBuildbasé, configurez les rôles du pipeline.](#)
 - Pour un [provisionnement autogéré](#), configurez un référentiel de [pipelines](#).

Pour plus d'informations sur les méthodes de provisionnement, consultez [the section called “AWS-appvisionnement géré”](#).

Configuration d'un compartiment Amazon S3

Pour configurer un compartiment S3, suivez les instructions de la [section Créez votre premier compartiment S3](#) pour configurer un compartiment S3. Placez vos entrées AWS Proton dans le compartiment où AWS Proton vous pourrez les récupérer. Ces entrées sont appelées ensembles de modèles. Vous pouvez en apprendre davantage à leur sujet dans d'autres sections de ce guide.

Configuration d'une AWS CodeStar connexion

Pour vous connecter AWS Proton à un référentiel, vous créez une AWS CodeStar connexion qui active un pipeline lorsqu'un nouveau commit est effectué sur un référentiel de code source tiers.

AWS Proton utilise la connexion pour :

- Activez un pipeline de services lorsqu'un nouveau commit est effectué sur le code source de votre référentiel.

- Effectuez une pull request sur une infrastructure en tant que référentiel de code.
- Créez une nouvelle version mineure ou majeure du modèle chaque fois qu'un commit est envoyé vers un référentiel de modèles qui modifie l'un de vos modèles, si la version n'existe pas déjà.

Vous pouvez vous connecter aux référentiels Bitbucket GitHub, GitHub Enterprise et GitHub Enterprise Server avec CodeConnections. Pour plus d'informations, consultez [CodeConnections](#) dans le Guide de l'utilisateur AWS CodePipeline.

Pour configurer une CodeStar connexion.

1. Ouvrez la [console AWS Proton](#).
2. Dans le volet de navigation, sélectionnez Paramètres, puis Connexions au référentiel pour accéder à la page Connexions dans les paramètres des outils de développement. La page affiche la liste des connexions.
3. Choisissez Créer une connexion et suivez les instructions.

Configuration des paramètres du pipeline CI/CD du compte

AWS Proton peut fournir des pipelines CI/CD pour déployer du code d'application dans vos instances de service. Les paramètres AWS Proton dont vous avez besoin pour le provisionnement du pipeline dépendent de la méthode de provisionnement que vous avez choisie pour votre pipeline.

AWS-provisionnement géré et CodeBuild basé : configuration des rôles de pipeline

Avec le [provisionnement et le provisionnement AWS gérés](#), [CodeBuild AWS Proton provisionnez](#) les pipelines pour vous. Par conséquent, AWS Proton nécessite un rôle de service fournissant des autorisations pour le provisionnement des pipelines. Chacune de ces deux méthodes de provisionnement utilise son propre rôle de service. Ces rôles sont partagés entre tous les pipelines de AWS Proton services et vous ne les configurez qu'une seule fois dans les paramètres de votre compte.

Pour créer des rôles de service de pipeline à l'aide de la console

1. Ouvrez la [console AWS Proton](#).
2. Dans le volet de navigation, choisissez Paramètres, puis Paramètres du compte.
3. Sur la page des paramètres CI/CD du compte, choisissez Configurer.
4. Effectuez l'une des actions suivantes :

- Pour avoir AWS Proton créé un rôle de service de pipeline pour vous

[Pour activer le provisionnement AWS géré des pipelines] Sur la page Configurer les paramètres du compte, dans la section du rôle du pipeline de provisionnement AWS géré :

- a. Sélectionnez Nouveau rôle de service.
- b. Entrez un nom pour le rôle, par exemple, **myProtonPipelineServiceRole**.
- c. Cochez la case pour accepter de créer un AWS Proton rôle doté de privilèges administratifs dans votre compte.

[Pour activer le provisionnement CodeBuild basé sur les pipelines] Sur la page Configurer les paramètres du compte, dans la section Rôle de CodeBuild pipeline, choisissez Rôle de service existant, puis choisissez le rôle de service que vous avez créé dans la section Rôle de CloudFormation pipeline. Ou, si vous n'avez attribué aucun rôle de CloudFormation pipeline, répétez les trois étapes précédentes pour créer un nouveau rôle de service.

- Pour choisir les rôles de service de pipeline existants

[Pour activer le provisionnement AWS géré des pipelines] Sur la page Configurer les paramètres du compte, dans la section rôle du pipeline de provisionnement AWS géré, choisissez Rôle de service existant, puis choisissez un rôle de service dans votre compte.
AWS

[Pour activer le CodeBuild provisionnement des pipelines] Sur la page Configurer les paramètres du compte, dans la section Rôle de provisionnement du CodeBuild pipeline, choisissez Rôle de service existant, puis choisissez un rôle de service dans votre AWS compte.

5. Choisissez Save Changes (Enregistrer les modifications).

Votre nouveau rôle de service de pipeline est affiché sur la page des paramètres du compte.


Approvisionnement autogéré : configurez un référentiel de pipelines

Avec le [provisionnement autogéré](#), AWS Proton envoie une pull request (PR) à un référentiel de provisionnement que vous avez configuré, et votre code d'automatisation est responsable du provisionnement des pipelines. Par conséquent, AWS Proton n'a pas besoin d'un rôle de service pour approvisionner des pipelines. Il a plutôt besoin d'un référentiel de provisioning enregistré. Votre

code d'automatisation dans le référentiel doit assumer un rôle approprié qui fournit des autorisations pour le provisionnement des pipelines.

Pour enregistrer un référentiel de provisionnement de pipelines à l'aide de la console

1. Créez un référentiel de provisionnement du pipeline CI/CD si vous n'en avez pas encore créé un. Pour plus d'informations sur les pipelines dans le cadre du provisionnement autogéré, consultez [the section called "Approvisionnement"](#)
2. Dans le volet de navigation, choisissez Paramètres, puis Paramètres du compte.
3. Sur la page des paramètres CI/CD du compte, choisissez Configurer.
4. Sur la page Configurer les paramètres du compte, dans la section du référentiel du pipeline CI/CD :
 - a. Sélectionnez Nouveau référentiel, puis choisissez l'un des fournisseurs de référentiels.
 - b. Pour la CodeStar connexion, choisissez l'une de vos connexions.

 Note

Si vous n'êtes pas encore connecté au compte du fournisseur de référentiel concerné, choisissez Ajouter une nouvelle CodeStar connexion, terminez le processus de création de connexion, puis cliquez sur le bouton d'actualisation situé à côté du menu de CodeStarconnexion. Vous devriez maintenant pouvoir choisir votre nouvelle connexion dans le menu.

- c. Dans Nom du référentiel, choisissez le référentiel de provisionnement de votre pipeline. Le menu déroulant affiche la liste des référentiels du compte fournisseur.
 - d. Dans Nom de la branche, choisissez l'une des branches du référentiel.
5. Choisissez Save Changes (Enregistrer les modifications).

Le référentiel de votre pipeline est affiché sur la page des paramètres du compte.

Configuration du AWS CLI

Pour utiliser le AWS CLI pour effectuer des appels d'AWS ProtonAPI, vérifiez que vous avez installé la dernière version du AWS CLI. Pour plus d'informations, consultez [Mise en route avec le AWS CLI](#) dans le AWS Command Line Interface Guide de l'utilisateur. Ensuite, pour commencer à utiliser le AWS CLI with AWS Proton, voir [the section called "Commencer à utiliser la CLI"](#).

Démarrer avec AWS Proton

Avant de commencer, [configurez-vous](#) pour utiliser AWS Proton et vérifiez que vous avez rempli les [conditions requises pour le démarrage](#).

Commencez AWS Proton par choisir un ou plusieurs des chemins suivants :

- Suivez un [exemple de flux de travail guidé par une console ou une CLI](#) via les liens de documentation.
- Exécutez un [exemple de flux de travail guidé sur console](#).
- Exécutez un [exemple de AWS CLI flux](#) de travail guidé.

Rubriques

- [Prérequis](#)
- [Flux de travail de démarrage](#)
- [Démarrer avec le kit AWS Management Console](#)
- [Démarrer avec le kit AWS CLI](#)
- [La bibliothèque AWS Proton de modèles](#)

Prérequis

Avant de commencer à l'utiliser AWS Proton, assurez-vous que les conditions préalables suivantes sont remplies. Pour plus d'informations, veuillez consulter [Configuration](#).

- Vous disposez d'un compte IAM avec des autorisations d'administrateur. Pour plus d'informations, veuillez consulter [Configuration avec IAM](#).
- Vous avez le rôle AWS Proton de service et le rôle de service de AWS Proton pipeline sont associés à votre compte. Pour plus d'informations, consultez [Configuration des rôles AWS Proton de service](#) et [Fonctions de service pour AWS Proton](#).
- Vous avez un AWS CodeStar lien. Pour plus d'informations, veuillez consulter [Configuration d'une AWS CodeStar connexion](#).
- Vous êtes habitué à créer des AWS CloudFormation modèles et à paramétrer Jinja. Pour plus d'informations, voir [Qu'est-ce que c'est AWS CloudFormation ?](#) dans le guide de l'AWS CloudFormation utilisateur et sur le [site Web de Jinja](#).

- Vous avez une connaissance pratique des services AWS d'infrastructure.
- Vous êtes connecté à votre Compte AWS.

Flux de travail de démarrage

Apprenez à créer des ensembles de modèles, à créer et à enregistrer des modèles, ainsi qu'à créer des environnements et des services en suivant les exemples d'étapes et les liens.

Avant de commencer, vérifiez que vous avez créé un [rôle AWS Proton de service](#).

Si votre modèle de service inclut un pipeline AWS Proton de service, vérifiez que vous avez créé une [AWS CodeStarconnexion](#) et un [rôle de service de AWS Proton pipeline](#).

Pour plus d'informations, consultez [The AWS Proton service API Reference](#).

Exemple : flux de travail de démarrage

1. Reportez-vous au schéma ci-dessous [Fonctionnement d'AWS Proton](#) pour une vue de haut niveau des AWS Proton entrées et des sorties.
2. [Créez un ensemble d'environnements et un ensemble de modèles de services](#).
 - a. Identifiez les [paramètres d'entrée](#).
 - b. Créez un [fichier de schéma](#).
 - c. Créez des [fichiers d'infrastructure sous forme de code \(IaC\)](#).
 - d. Pour [compléter votre ensemble de modèles, créez](#) un fichier manifeste et organisez vos fichiers IaC, vos fichiers manifestes et vos fichiers de schéma dans des répertoires.
 - e. Rendez votre [ensemble de modèles](#) accessible à AWS Proton.
3. [Créez et enregistrez une version de modèle d'environnement](#) avec AWS Proton.

Lorsque vous utilisez la console pour créer et enregistrer un modèle, une version du modèle est automatiquement créée.

Lorsque vous utilisez le AWS CLI pour créer et enregistrer un modèle :

- a. Créez un modèle d'environnement.
- b. Créez une version du modèle d'environnement.

Pour plus d'informations, consultez [CreateEnvironmentTemplate](#) et [CreateEnvironmentTemplateVersion](#) dans la référence de AWS Proton l'API.

4. [Publiez votre modèle d'environnement](#) pour le rendre utilisable.

Pour plus d'informations, consultez [UpdateEnvironmentTemplateVersion](#) la référence de AWS Proton l'API.

5. Pour [créer un environnement](#), sélectionnez une version de modèle d'environnement publiée et fournissez des valeurs pour les entrées requises.

Pour plus d'informations, consultez [CreateEnvironment](#) la référence de AWS Proton l'API.

6. [Créez et enregistrez une version de modèle de service](#) avec AWS Proton.

Lorsque vous utilisez la console pour créer et enregistrer un modèle, une version du modèle est automatiquement créée.

Lorsque vous utilisez le AWS CLI pour créer et enregistrer un modèle :

- a. Créez un modèle de service.
- b. Créez une version de modèle de service.

Pour plus d'informations, consultez [CreateServiceTemplate](#) et [CreateServiceTemplateVersion](#) dans la référence de AWS Proton l'API.

7. [Publiez votre modèle de service](#) pour le rendre utilisable.

Pour plus d'informations, consultez [UpdateServiceTemplateVersion](#) la référence de AWS Proton l'API.

8. Pour [créer un service](#), sélectionnez une version de modèle de service publiée et fournissez des valeurs pour les entrées requises.

Pour plus d'informations, consultez [CreateService](#) la référence de AWS Proton l'API.

Démarrer avec le kit AWS Management Console

Premiers pas avec AWS Proton

- Créez et visualisez un modèle d'environnement.

- Créez, visualisez et publiez un modèle de service qui utilise le modèle d'environnement que vous venez de créer.
- Créez un environnement et un service (facultatif).
- Supprimez le modèle de service, le modèle d'environnement, l'environnement et le service, s'ils ont été créés.

Étape 1 : ouvrir la AWS Proton console

- Ouvrez la [console AWS Proton](#).

Étape 2 : Préparez-vous à utiliser les modèles d'exemple

1. Créez une connexion Codestar à Github et nommez la connexion. my-proton-connection
2. Accédez à <https://github.com/aws-samples/aws-proton-cloudformation-sample-templates>
3. Créez un fork du dépôt dans votre compte Github.

Étape 3 : Création d'un modèle d'environnement

Dans le volet de navigation, sélectionnez Modèles d'environnement.

1. Sur la page Modèles d'environnement, choisissez Créer un modèle d'environnement.
2. Sur la page Créer un modèle d'environnement, dans la section Options du modèle, choisissez Créer un modèle pour le provisionnement de nouveaux environnements.
3. Dans la section Source du bundle de modèles, choisissez Synchroniser un ensemble de modèles depuis Git.
4. Dans la section Référentiel de définition du modèle, sélectionnez Choisir un dépôt Git lié.
5. Sélectionnez dans my-proton-connection la liste des référentiels.
6. Sélectionnez principal dans la liste des branches.
7. Dans la section des détails du modèle d'environnement Proton.
 - a. Entrez le nom du modèle sous la forme **fargate-env**.
 - b. Entrez le nom d'affichage du modèle d'environnement sous la forme **My Fargate Environment**.
 - c. (Facultatif) Entrez une description pour le modèle d'environnement.

8. (Facultatif) Dans la section Tags, choisissez Ajouter un nouveau tag et entrez une clé et une valeur pour créer un tag géré par le client.
9. Choisissez Créer un modèle d'environnement.

Vous êtes maintenant sur une nouvelle page qui affiche le statut et les détails de votre nouveau modèle d'environnement. Ces informations incluent une liste de balises gérées par le client AWS et une liste de balises gérées par le client. AWS Proton génère automatiquement des balises AWS gérées pour vous lorsque vous créez des AWS Proton ressources. Pour plus d'informations, veuillez consulter [AWS Proton ressources et balises](#).

10. Le statut d'un nouveau modèle d'environnement commence à l'état Brouillon. Vous et les autres personnes `proton:CreateEnvironment` autorisées pouvez le consulter et y accéder. Suivez l'étape suivante pour mettre le modèle à la disposition des autres utilisateurs.
11. Dans la section Versions du modèle, cliquez sur le bouton radio situé à gauche de la version mineure du modèle que vous venez de créer (1.0). Vous pouvez également choisir Publier dans la bannière d'alerte d'information et ignorer l'étape suivante.
12. Dans la section Versions du modèle, choisissez Publier.
13. Le statut du modèle passe à Publié. Comme il s'agit de la dernière version du modèle, il s'agit de la version recommandée.
14. Dans le volet de navigation, sélectionnez Modèles d'environnement.

Une nouvelle page affiche la liste de vos modèles d'environnement ainsi que les détails des modèles.

Étape 4 : Création d'un modèle de service

Créez un modèle de service.

1. Dans le volet de navigation, sélectionnez Modèles de services.
2. Sur la page Modèles de service, choisissez Créer un modèle de service.
3. Sur la page Créer un modèle de service, dans la section Source du bundle de modèles, choisissez Synchroniser un ensemble de modèles depuis Git.
4. Dans la section Modèle, sélectionnez Choisir un dépôt Git lié.
5. Sélectionnez dans `my-proton-connection` la liste des référentiels.
6. Sélectionnez principal dans la liste des branches.
7. Dans la section des détails du modèle de service Proton.

- a. Entrez le nom du modèle de service sous la forme **backend-fargate-svc**.
 - b. Entrez le nom d'affichage du modèle de service sous la forme **My Fargate Service**.
 - c. (Facultatif) Entrez une description pour le modèle de service.
8. Dans la section Modèles d'environnement compatibles.
- Cochez la case située à gauche du modèle d'environnement My Fargate Environment pour sélectionner le modèle d'environnement compatible pour le nouveau modèle de service.
9. Pour les paramètres de chiffrement, conservez les valeurs par défaut.
10. Dans la section Définition du pipeline.
- Conservez le bouton Ce modèle inclut un pipeline CI/CD sélectionné.
11. Choisissez Créer un modèle de service.

Vous êtes maintenant sur une nouvelle page qui affiche le statut et les détails de votre nouveau modèle de service, y compris une liste de balises gérées par le client AWS et des balises gérées par le client.

12. Le statut d'un nouveau modèle de service commence à l'état Brouillon. Seuls les administrateurs peuvent le consulter et y accéder. Pour mettre le modèle de service à la disposition des développeurs, procédez à l'étape suivante.
13. Dans la section Versions du modèle, cliquez sur le bouton radio situé à gauche de la version mineure du modèle que vous venez de créer (1.0). Vous pouvez également choisir Publier dans la bannière d'alerte d'information et ignorer l'étape suivante.
14. Dans la section Versions du modèle, choisissez Publier.
15. Le statut du modèle passe à Publié.

La première version mineure de votre modèle de service est publiée et peut être utilisée par les développeurs. Comme il s'agit de la dernière version du modèle, il s'agit de la version recommandée.

16. Dans le volet de navigation, sélectionnez Modèles de services.

Une nouvelle page affiche la liste de vos modèles de services et de leurs détails.

Étape 5 : Création d'un environnement

Dans le panneau de navigation, choisissez Environments (Environnements).

1. Choisissez Create environment.
2. Sur la page Choisir un modèle d'environnement, sélectionnez le modèle que vous venez de créer. Il s'appelle My Fargate Environment. Choisissez ensuite Configurer.
3. Sur la page Configurer l'environnement, dans la section Provisioning, choisissez Provisioning through AWS Proton.
4. Dans la section Compte de déploiement, sélectionnez Ceci Compte AWS.
5. Dans Paramètres d'environnement, entrez le nom de l'environnement sous la forme **my-fargate-environment**.
6. Dans la section Rôles environnementaux, sélectionnez Nouveau rôle de service ou, si vous avez déjà créé un rôle de AWS Proton service, sélectionnez Rôle de service existant.
 - a. Sélectionnez Nouveau rôle de service pour créer un nouveau rôle.
 - i. Entrez le nom du rôle d'environnement sous la forme **MyProtonServiceRole**.
 - ii. Cochez la case pour accepter de créer un rôle AWS Proton de service avec des privilèges administratifs pour votre compte.
 - b. Sélectionnez Rôle de service existant pour utiliser un rôle existant.
 - Sélectionnez votre rôle dans le champ déroulant Nom du rôle environnemental.
7. Choisissez Suivant.
8. Sur la page Configurer les paramètres personnalisés, utilisez les valeurs par défaut.
9. Choisissez Next et passez en revue vos entrées.
10. Sélectionnez Create (Créer).

Consultez les détails et l'état de l'environnement, ainsi que les balises AWS gérées et les balises gérées par le client pour votre environnement.

11. Dans le panneau de navigation, choisissez Environments (Environnements).

Une nouvelle page affiche la liste de vos environnements ainsi que le statut et d'autres détails relatifs à l'environnement.

Étape 6 : Facultatif - Création d'un service et déploiement d'une application

1. Ouvrez la [console AWS Proton](#).
2. Dans le volet de navigation, sélectionnez Services.

3. Sur la page Services, choisissez Créer un service.
4. Sur la page Choisir un modèle de service, sélectionnez le modèle My Fargate Service en cliquant sur le bouton radio situé dans le coin supérieur droit de la fiche modèle.
5. Choisissez Configurer dans le coin inférieur droit de la page.
6. Sur la page Configurer le service, dans la section Paramètres du service, entrez le nom du service **my-service**.
7. (Facultatif) Entrez une description du service.
8. Dans la section Paramètres du référentiel de services :
 - a. Pour CodeStar la connexion, choisissez votre connexion dans la liste.
 - b. Pour Nom du référentiel, choisissez le nom de votre référentiel de code source dans la liste.
 - c. Pour Nom de la branche, choisissez le nom de la branche de votre référentiel de code source dans la liste.
9. (Facultatif) Dans la section Tags, choisissez Ajouter un nouveau tag et entrez une clé et une valeur pour créer un tag géré par le client. Sélectionnez ensuite Next (Suivant).
10. Sur la page Configurer les paramètres personnalisés, dans la section Instances de service, dans la section Nouvelle instance, suivez les étapes suivantes pour fournir des valeurs personnalisées pour les paramètres de vos instances de service.
 - a. Entrez le nom de l'instance **my-app-service**.
 - b. Choisissez l'environnement **my-fargate-environment** de votre instance de service.
 - c. Conservez les valeurs par défaut pour les paramètres d'instance restants.
 - d. Conservez les valeurs par défaut pour les entrées du pipeline.
 - e. Choisissez Next et passez en revue vos entrées.
 - f. Choisissez Créer et consultez l'état et les détails de votre service.
11. Sur la page des détails du service, consultez l'état de votre instance de service et de votre pipeline en choisissant les onglets Vue d'ensemble et Pipeline. Sur ces pages, vous pouvez également consulter les AWS tags gérés par les clients. AWS Proton crée automatiquement des tags AWS gérés pour vous. Choisissez Gérer les balises pour créer et modifier les balises gérées par le client. Pour plus d'informations sur le balisage, consultez [AWS Proton ressources et balises](#).
12. Une fois le service actif, dans l'onglet Vue d'ensemble, dans la section Instances de service, choisissez le nom de votre instance de service my-app-service.

Vous êtes maintenant sur la page détaillée de l'instance de service.

13. Pour afficher votre application, dans la section Sorties, copiez le ServiceEndpointlien dans votre navigateur.

Vous voyez un AWS Proton graphique sur la page Web.

14. Une fois le service créé, dans le volet de navigation, choisissez Services pour afficher la liste de vos services.

Étape 7 : Nettoyez.

1. Ouvrez la [console AWS Proton](#).
2. Supprimer un service (si vous en avez créé un)
 - a. Dans le volet de navigation, sélectionnez Services.
 - b. Sur la page Services, choisissez le nom du service my-service.

Vous êtes maintenant sur la page détaillée du service de my-service.

- c. Dans le coin supérieur droit de la page, choisissez Actions, puis Supprimer.
 - d. Un modal vous invite à confirmer l'action de suppression.
 - e. Suivez les instructions et choisissez Oui, supprimer.
3. Supprimer un environnement
 - a. Dans le panneau de navigation, choisissez Environnements (Environnements).
 - b. Sur la page Environnements, sélectionnez le bouton radio situé à gauche de l'environnement que vous venez de créer.
 - c. Choisissez Actions, puis Supprimer.
 - d. Un modal vous invite à confirmer l'action de suppression.
 - e. Suivez les instructions et choisissez Oui, supprimer.
 4. Supprimer un modèle de service
 - a. Dans le volet de navigation, sélectionnez Modèles de services.
 - b. Sur la page Modèles de service, sélectionnez le bouton radio situé à gauche du modèle de service my-svc-template.
 - c. Choisissez Actions, puis Supprimer.

- d. Un modal vous invite à confirmer l'action de suppression.
 - e. Suivez les instructions et choisissez Oui, supprimer. Cela supprime le modèle de service et toutes ses versions.
5. Supprimer un modèle d'environnement
 - a. Dans le volet de navigation, sélectionnez Modèles d'environnement.
 - b. Sur la page Modèles d'environnement, sélectionnez le bouton radio situé à gauche de my-env-template.
 - c. Choisissez Actions, puis Supprimer.
 - d. Un modal vous invite à confirmer l'action de suppression.
 - e. Suivez les instructions et choisissez Oui, supprimer. Cela supprime le modèle d'environnement et toutes ses versions.
 6. Supprimer votre connexion Codestar

Démarrer avec le kit AWS CLI

Pour commencer à AWS Proton utiliser le AWS CLI, suivez ce didacticiel. Le didacticiel présente un AWS Proton service d'équilibrage de charge destiné au public basé sur AWS Fargate. Le didacticiel fournit également un pipeline CI/CD qui déploie un site Web statique avec une image affichée.

Avant de commencer, assurez-vous que vous êtes correctement configuré. Pour plus de détails, consultez [the section called "Prérequis"](#).

Étape 1 : enregistrer un modèle d'environnement

Au cours de cette étape, en tant qu'administrateur, vous enregistrez un exemple de modèle d'environnement, qui contient un cluster Amazon Elastic Container Service (Amazon ECS) et un Amazon Virtual Private Cloud (Amazon VPC) avec deux sous-réseaux publics/privés.

Pour enregistrer un modèle d'environnement

1. Intégrez le référentiel [AWS Proton d'exemples CloudFormation de modèles](#) à votre GitHub compte ou à votre organisation. Ce référentiel inclut les modèles d'environnement et de service que nous utilisons dans ce didacticiel.

Enregistrez ensuite votre dépôt bifurqué auprès AWS Proton de. Pour plus d'informations, veuillez consulter [the section called "Création d'un lien vers le référentiel"](#).

2. Créez un modèle d'environnement.

La ressource du modèle d'environnement suit les versions du modèle d'environnement.

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

3. Créez une configuration de synchronisation de modèles.

AWS Proton définit une relation de synchronisation entre votre référentiel et votre modèle d'environnement. Il crée ensuite la version 1.0 du modèle dans DRAFT status.

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

4. Attendez que la version du modèle d'environnement soit correctement enregistrée.

Lorsque cette commande revient avec un statut de sortie de 0, l'enregistrement de la version est terminé. Cela est utile dans les scripts pour garantir que vous pouvez exécuter correctement la commande à l'étape suivante.

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

5. Publiez la version du modèle d'environnement pour la rendre disponible pour la création de l'environnement.

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

Étape 2 : enregistrer un modèle de service

Au cours de cette étape, en tant qu'administrateur, vous enregistrez un exemple de modèle de service, qui contient toutes les ressources nécessaires pour fournir un service Amazon ECS Fargate via un équilibreur de charge et un pipeline CI/CD qui utilise AWS CodePipeline

Pour enregistrer un modèle de service

1. Créez un modèle de service.

La ressource de modèle de service suit les versions des modèles de service.

```
$ 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. Créez une configuration de synchronisation de modèles.

AWS Proton définit une relation de synchronisation entre votre référentiel et votre modèle de service. Il crée ensuite la version 1.0 du modèle dans DRAFT status.

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

3. Attendez que la version du modèle de service soit correctement enregistrée.

Lorsque cette commande revient avec un statut de sortie de 0, l'enregistrement de la version est terminé. Cela est utile dans les scripts pour garantir que vous pouvez exécuter correctement la commande à l'étape suivante.

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

4. Publiez la version du modèle de service pour la rendre disponible pour la création du service.

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

Étape 3 : Déployer un environnement

Au cours de cette étape, en tant qu'administrateur, vous instanciez un AWS Proton environnement à partir du modèle d'environnement.

Pour déployer un environnement

1. Obtenez un exemple de fichier de spécifications pour le modèle d'environnement que vous avez enregistré.

Vous pouvez télécharger le fichier `environment-templates/fargate-env/spec/spec.yaml` depuis le référentiel d'exemples de modèles. Vous pouvez également récupérer l'intégralité du référentiel localement et exécuter la `create-environment` commande depuis le `environment-templates/fargate-env` répertoire.

2. Créez un environnement.

AWS Proton lit les valeurs d'entrée de votre spécification d'environnement, les combine avec votre modèle d'environnement et provisionne les ressources environnementales de votre AWS compte en utilisant votre rôle de AWS Proton service.

```
$ 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/AWSProtonServiceRole" \  
  --spec "file:///spec/spec.yaml"
```

3. Attendez que l'environnement soit correctement déployé.

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


Étape 4 : Déployer un service [développeur d'applications]

Au cours des étapes précédentes, un administrateur a enregistré et publié un modèle de service et a déployé un environnement. En tant que développeur d'applications, vous pouvez désormais créer un AWS Proton service et le déployer dans l'AWS Proton environnement

Pour déployer un service

1. Obtenez un exemple de fichier de spécifications pour le modèle de service enregistré par l'administrateur.

Vous pouvez télécharger le fichier `service-templates/load-balanced-fargate-svc/spec/spec.yaml` depuis le référentiel d'exemples de modèles. Vous pouvez également récupérer l'intégralité du référentiel localement et exécuter la `create-service` commande depuis le `service-templates/load-balanced-fargate-svc` répertoire.

2. Intégrez le référentiel [AWS Proton Sample Services](#) à votre GitHub compte ou à votre organisation. Ce référentiel inclut le code source de l'application que nous utilisons dans ce didacticiel.
3. Créer un service.

AWS Proton lit les valeurs d'entrée de votre spécification de service, les combine avec votre modèle de service et fournit les ressources de service de votre AWS compte dans l'environnement spécifié dans la spécification. Un AWS CodePipeline pipeline déploie le code de votre application à partir du référentiel que vous spécifiez dans la commande.

```
$ 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. Attendez que le service soit correctement déployé.

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

5. Récupérez les résultats et consultez votre nouveau site Web.

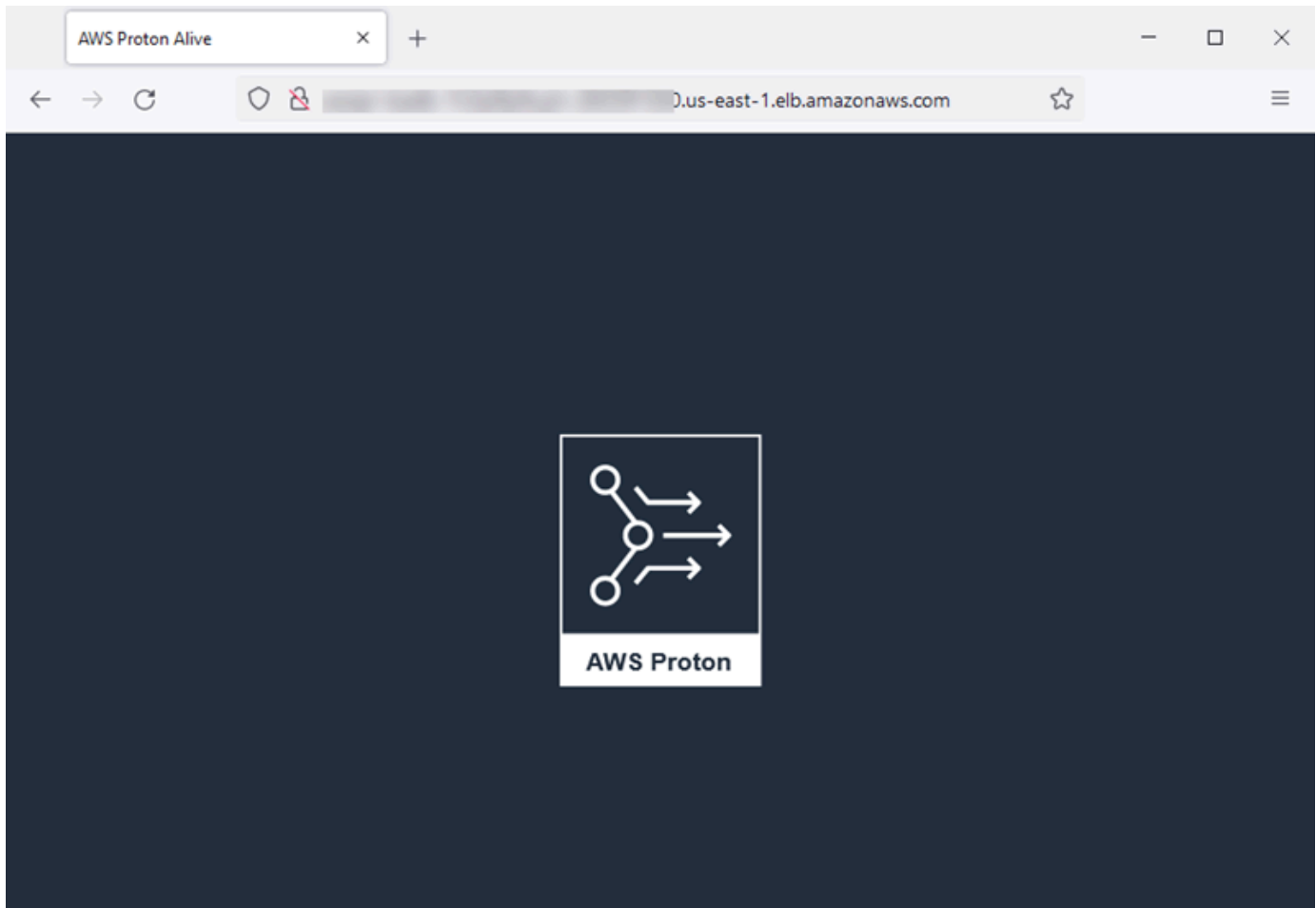
Exécutez la commande suivante :

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

Le résultat de la commande doit être similaire à ce qui suit :

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

La valeur de la sortie de l'`ServiceURL` instance est le point de terminaison de votre nouveau site Web de service. Utilisez votre navigateur pour y accéder. Vous devriez voir le graphique suivant sur une page statique :



Étape 5 : Nettoyage (facultatif)

Au cours de cette étape, lorsque vous avez terminé d'explorer les AWS ressources que vous avez créées dans le cadre de ce didacticiel, et pour économiser sur les coûts associés à ces ressources, vous les supprimez.

Pour supprimer les ressources du didacticiel

1. Pour supprimer le service, exécutez la commande suivante :

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

2. Pour supprimer l'environnement, exécutez la commande suivante :

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

3. Pour supprimer le modèle de service, exécutez les commandes suivantes :

```
$ 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. Pour supprimer le modèle d'environnement, exécutez les commandes suivantes :

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

La bibliothèque AWS Proton de modèles

L'AWS Proton équipe gère une bibliothèque d'exemples de modèles sur GitHub. La bibliothèque inclut des exemples de fichiers d'infrastructure sous forme de code (IaC) pour de nombreux scénarios courants d'environnement et d'infrastructure d'applications.

La bibliothèque de modèles est stockée dans deux GitHub référentiels :

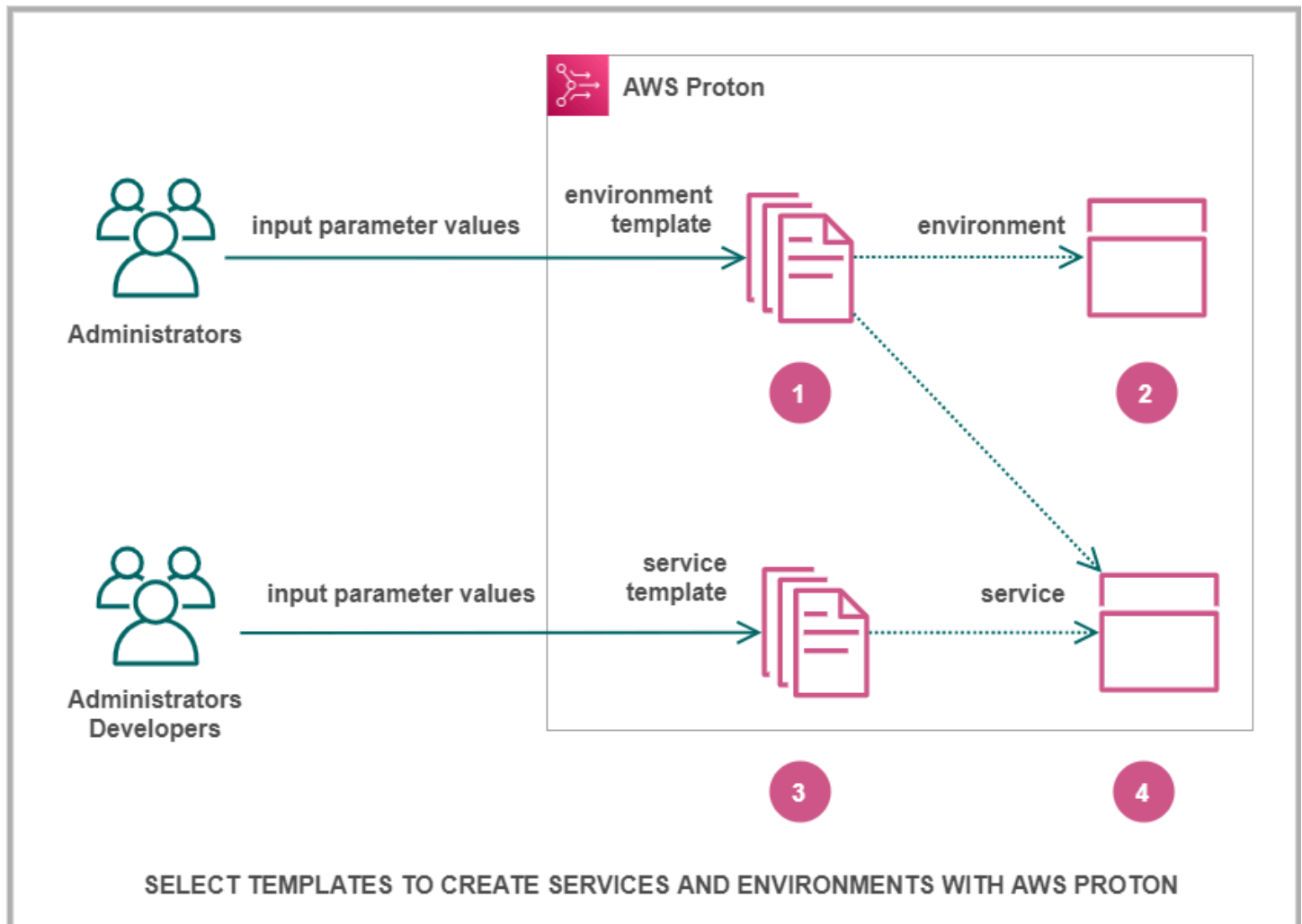
- [aws-proton-cloudformation-sample-templates](#) — Exemples de ensembles de modèles qui utilisent AWS CloudFormation Jinja comme langage IaC. Vous pouvez utiliser ces exemples pour les [AWS-approvisionnement géré](#) environnements.
- [aws-proton-terraform-sample-templates](#) — Exemples de bundles de modèles qui utilisent Terraform comme langage IaC. Vous pouvez utiliser ces exemples pour les [Approvisionnement](#) environnements.

Chacun de ces référentiels possède un README fichier contenant des informations complètes sur le contenu et la structure du référentiel. Chaque exemple contient des informations sur le cas d'utilisation couvert par le modèle, son architecture et les paramètres d'entrée utilisés par le modèle.

Vous pouvez utiliser les modèles de cette bibliothèque directement en connectant l'un des référentiels de la bibliothèque à votre GitHub compte. Vous pouvez également utiliser ces exemples comme point de départ pour développer votre environnement et vos modèles de services.

Fonctionnement d'AWS Proton

Avec AWS Proton, vous provisionnez des environnements, puis des services qui s'exécutent dans ces environnements. Les environnements et les services sont basés sur des modèles d'environnement et de service, respectivement, que vous choisissez dans votre bibliothèque de modèles AWS Proton versionnés.



1

Lorsqu vous, en tant qu'administrateur, sélectionnez un modèle d'environnement avec AWS Proton, vous fournissez des valeurs pour les paramètres d'entrée requis.

2

AWS Proton utilise le modèle d'environnement et les valeurs des paramètres pour provisionner votre environnement.

3

vous, en tant que développeur ou administrateur, sélectionnez un modèle de service avec AWS Proton, vous fournissez des valeurs pour les paramètres d'entrée requis. Vous sélectionnez également l'environnement dans lequel déployer votre application ou votre service.

Lorsqu

4

Proton utilise le modèle de service, ainsi que les valeurs de votre service et des paramètres d'environnement sélectionnés, pour fournir votre service.

AWS

Vous fournissez des valeurs pour les paramètres d'entrée afin de personnaliser votre modèle en vue de le réutiliser et de multiples cas d'utilisation, applications ou services.

Pour que cela fonctionne, vous créez des ensembles de modèles d'environnement ou de service et vous les téléchargez dans des modèles d'environnement ou de service enregistrés, respectivement.

[Les ensembles de modèles](#) contiennent tout ce dont vous AWS Proton avez besoin pour fournir des environnements ou des services.

Lorsque vous créez un modèle d'environnement ou de service, vous chargez un ensemble de modèles qui contient l'infrastructure paramétrée sous forme de fichiers de code (IaC) AWS Proton utilisés pour fournir des environnements ou des services.

Lorsque vous sélectionnez un modèle d'environnement ou de service pour créer ou mettre à jour un environnement ou un service, vous fournissez des valeurs pour les paramètres du fichier IaC du bundle de modèles.

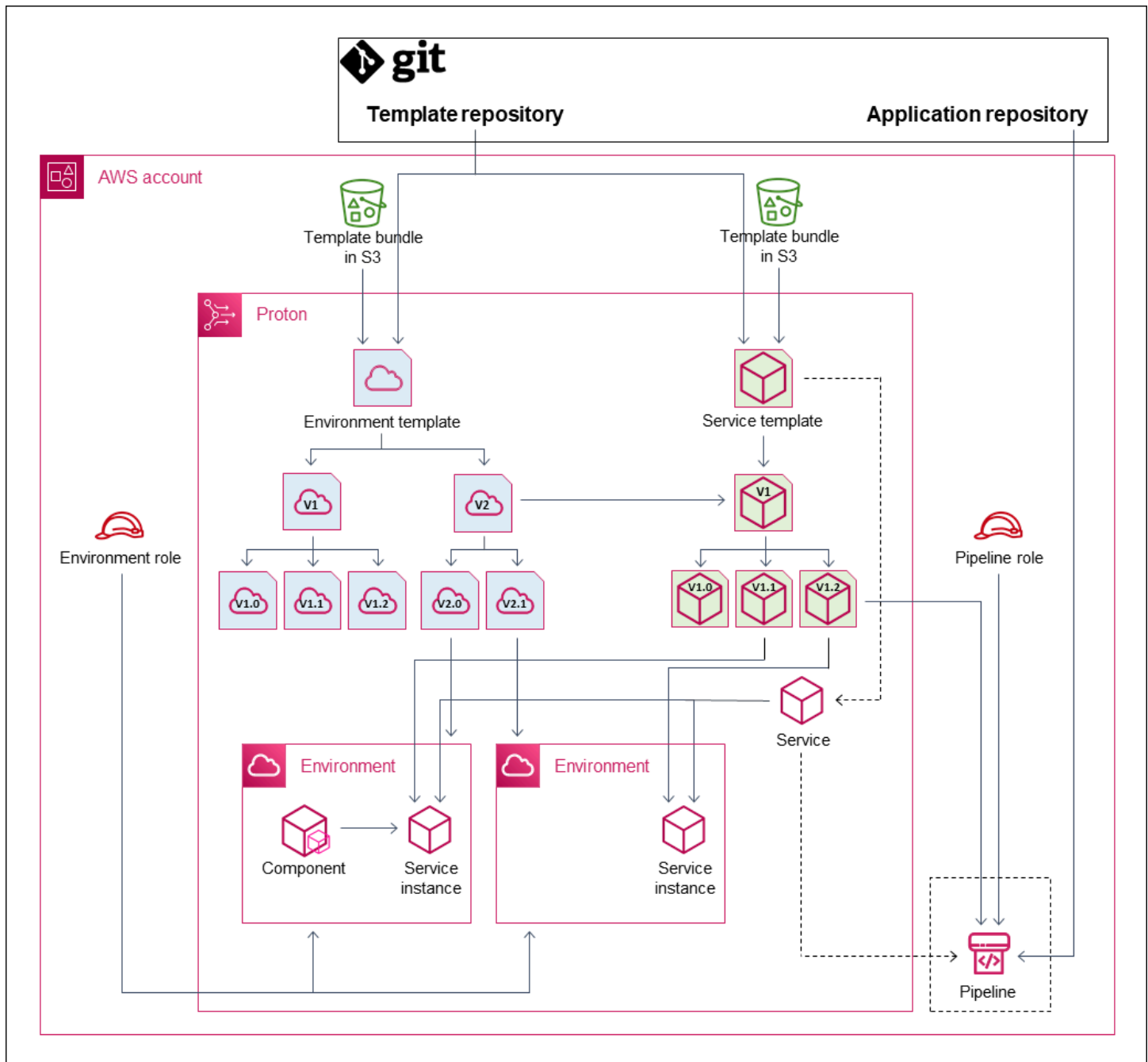
Rubriques

- [AWS Proton objets](#)
- [Comment AWS Proton approvisionne l'infrastructure](#)
- [Terminologie AWS Proton](#)

AWS Proton objets

Le schéma suivant montre les principaux AWS Proton objets et leurs relations avec d'autres objets AWS et des objets tiers. Les flèches représentent le sens du flux de données (sens inverse de la dépendance).

Nous suivons le schéma avec de brèves descriptions et des liens de référence pour ces AWS Proton objets.



- **Modèle d'environnement :** collection de versions de modèles d'environnement pouvant être utilisées pour créer AWS Proton des environnements.

Pour plus d'informations, consultez [Création de modèles et d'ensembles](#) et [Modèles](#).

- **Version du modèle d'environnement :** version spécifique d'un modèle d'environnement. Prend un bundle de modèles en entrée, soit à partir d'un bucket S3, soit à partir d'un référentiel Git. Le

bundle spécifie l'infrastructure sous forme de code (IaC) et les paramètres d'entrée associés pour un AWS Proton environnement.

Pour plus d'informations, consultez [the section called “Versions”](#), [the section called “Publier”](#) et [the section called “Configurations de synchronisation de modèles”](#).

- Environnement : ensemble de ressources d'AWS infrastructure partagées et de politiques d'accès dans AWS Proton auxquelles les services sont déployés. AWS les ressources sont provisionnées à l'aide d'une version du modèle d'environnement invoquée avec des valeurs de paramètres spécifiques. Les politiques d'accès sont fournies dans le cadre d'un rôle de service.

Pour plus d'informations, veuillez consulter [Environnements](#).

- Modèle de service : collection de versions de modèles de service pouvant être utilisées pour créer des AWS Proton services.

Pour plus d'informations, consultez [Création de modèles et d'ensembles](#) et [Modèles](#).

- Version du modèle de service : version spécifique d'un modèle de service. Prend un bundle de modèles en entrée, soit à partir d'un bucket S3, soit à partir d'un référentiel Git. Le bundle spécifie l'infrastructure sous forme de code (IaC) et les paramètres d'entrée associés pour un AWS Proton service.

Une version de modèle de service spécifie également ces contraintes sur les instances de service en fonction de la version :

- Modèles d'environnement compatibles : les instances peuvent uniquement s'exécuter dans des environnements basés sur ces modèles d'environnement compatibles.
- Sources de composants prises en charge : types de composants que les développeurs peuvent associer à des instances.

Pour plus d'informations, consultez [the section called “Versions”](#), [the section called “Publier”](#) et [the section called “Configurations de synchronisation de modèles”](#).

- Service : ensemble d'instances de service qui exécutent une application à l'aide des ressources spécifiées dans un modèle de service, et éventuellement un pipeline CI/CD qui déploie le code de l'application dans ces instances.

Dans le diagramme, la ligne pointillée du modèle de service signifie que le service transmet le modèle aux instances de service et au pipeline.

Pour plus d'informations, veuillez consulter [Services](#).

- **Instance de service** : ensemble de ressources d'AWS Infrastructure qui exécutent une application dans un AWS Proton environnement spécifique. Les ressources sont provisionnées à l'aide d'une version de modèle de service invoquée avec des valeurs de paramètres spécifiques.

Pour plus d'informations, consultez [Services](#) et [the section called "Mise à jour d'instance"](#).

- **Pipeline** : pipeline CI/CD optionnel qui déploie une application dans les instances d'un service, avec des politiques d'accès pour provisionner ce pipeline. Les politiques d'accès sont fournies dans le cadre d'un rôle de service. Un service n'est pas toujours associé à un AWS Proton pipeline. Vous pouvez choisir de gérer les déploiements de votre code d'application en dehors de celui-ci.

Dans le diagramme, la ligne pointillée représentant Service et la zone en pointillés autour de Pipeline indiquent que si vous choisissez de gérer vous-même vos déploiements CI/CD, le AWS Proton pipeline ne sera peut-être pas créé et votre propre pipeline ne se trouve peut-être pas dans votre compte. AWS

Pour plus d'informations, consultez [Services](#) et [the section called "Mise à jour du pipeline"](#).

- **Composant** : extension d'une instance de service définie par le développeur. Spécifie les ressources d'AWS Infrastructure supplémentaires dont une application particulière peut avoir besoin, en plus des ressources fournies par l'environnement et l'instance de service. Les équipes de plateforme contrôlent l'infrastructure qu'un composant peut fournir en attribuant un rôle de composant à l'environnement.

Pour plus d'informations, veuillez consulter [Composants](#).

Comment AWS Proton provisionne l'infrastructure

AWS Proton peut fournir une infrastructure de plusieurs manières :

- **AWS-provisionnement géré** : AWS Proton appelle le moteur de provisionnement en votre nom. Cette méthode ne prend en charge que les ensembles AWS CloudFormation de modèles. Pour plus d'informations, veuillez consulter [the section called "AWS CloudFormation fichiers IaC"](#).
- **CodeBuild provisionnement** : permet AWS Proton AWS CodeBuild d'exécuter les commandes shell que vous fournissez. Vos commandes peuvent lire les entrées que AWS Proton fournit et sont responsables du provisionnement ou du déprovisionnement de l'infrastructure et de la génération de valeurs de sortie. Un ensemble de modèles pour cette méthode inclut vos commandes dans un

fichier manifeste et tous les programmes, scripts ou autres fichiers dont ces commandes peuvent avoir besoin.

À titre d'exemple d'utilisation du CodeBuild provisionnement, vous pouvez inclure du code qui utilise le AWS Cloud Development Kit (AWS CDK) pour provisionner AWS des ressources, ainsi qu'un manifeste qui installe le CDK et exécute votre code CDK.

Pour plus d'informations, veuillez consulter [the section called “CodeBuild offre groupée”](#).

Note

Vous pouvez utiliser le CodeBuild provisionnement avec des environnements et des services. Pour le moment, vous ne pouvez pas provisionner des composants de cette manière.

- Provisionnement autogéré : AWS Proton émet une pull request (PR) vers un référentiel que vous fournissez, dans lequel votre propre système de déploiement d'infrastructure exécute le processus de provisionnement. Cette méthode ne prend en charge que les bundles de modèles Terraform. Pour plus d'informations, veuillez consulter [the section called “Fichiers Terraform iAC”](#).

AWS Proton détermine et définit la méthode de provisionnement pour chaque environnement et service séparément. Lorsque vous créez ou mettez à jour un environnement ou un service, AWS Proton examine le bundle de modèles que vous fournissez et détermine la méthode de provisionnement indiquée par le bundle de modèles. Au niveau de l'environnement, vous fournissez les paramètres dont l'environnement et ses services potentiels peuvent avoir besoin pour leurs méthodes de provisionnement : rôles AWS Identity and Access Management (IAM), connexion à un compte d'environnement ou référentiel d'infrastructure.

Les développeurs qui fournissent AWS Proton un service bénéficient de la même expérience, quelle que soit la méthode de provisionnement utilisée. Les développeurs n'ont pas besoin de connaître la méthode de provisionnement et n'ont pas besoin de modifier quoi que ce soit dans le processus de provisionnement des services. Le modèle de service définit la méthode de provisionnement et chaque environnement dans lequel un développeur déploie le service fournit les paramètres nécessaires au provisionnement des instances de service.

Le schéma suivant résume certaines caractéristiques principales des différentes méthodes de provisionnement. Les sections qui suivent le tableau fournissent des détails sur chaque méthode.

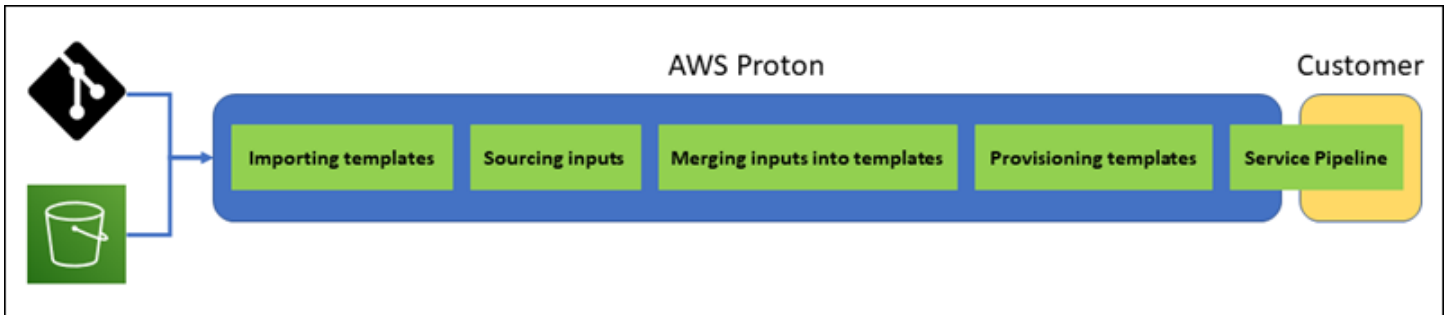
Méthode d'approvisionnement	Modèles	Approvisionnement	État suivi
AWS-géré	manifeste, schéma, fichier IaC (CloudFormation)	AWS Proton(à traversCloudFormation)	AWS Proton(à traversCloudFormation)
CodeBuild	manifeste (avec commandes), schéma, dépendances de commandes (par exemple, AWS CDK code)	AWS Proton(à traversCodeBuild)	AWS Proton(vos commandes renvoient l'état viaCodeBuild)
autogéré	manifeste, schéma, fichiers IaC (Terraform)	Votre code (via des actions Git)	Votre code (transmis AWS via un appel d'API)

Comment fonctionne AWS le provisionnement géré

Lorsqu'un environnement ou un service utilise le provisionnement AWS géré, l'infrastructure est provisionnée comme suit :

1. Un AWS Proton client (un administrateur ou un développeur) crée la AWS Proton ressource (un environnement ou un service). Le client sélectionne un modèle pour la ressource et fournit les paramètres requis. Pour plus d'informations, consultez la section suivante [the section called “Considérations relatives au AWS provisionnement géré”](#).
2. AWS Proton affiche un AWS CloudFormation modèle complet pour le provisionnement de la ressource.
3. AWS Proton appelle AWS CloudFormation pour démarrer le provisionnement à l'aide du modèle rendu.
4. AWS Proton surveille en permanence le AWS CloudFormation déploiement.
5. Une fois le provisionnement terminé, AWS Proton signale les erreurs en cas d'échec et capture les résultats du provisionnement, tels que l'identifiant Amazon VPC, en cas de succès.

Le schéma suivant montre qu'il AWS Proton prend directement en charge la plupart de ces étapes.



Considérations relatives au AWS provisionnement géré

- Rôle de provisionnement de l'infrastructure : lorsqu'un environnement ou l'une des instances de service qui s'y exécutent peut utiliser le provisionnement AWS géré, un administrateur doit configurer un rôle IAM (directement ou dans le cadre d'une connexion à un compte d'AWS Proton environnement). AWS Proton utilise ce rôle pour provisionner l'infrastructure de ces ressources AWS de provisionnement gérées. Le rôle doit disposer des autorisations nécessaires AWS CloudFormation pour créer toutes les ressources incluses dans les modèles de ces ressources.

Pour plus d'informations, consultez [the section called “Rôles IAM”](#) et [the section called “Exemples de politiques relatives aux rôles de service”](#).

- Provisionnement de services : lorsqu'un développeur déploie une instance de service qui utilise le AWS provisionnement géré dans l'environnement, AWS Proton utilise le rôle attribué à cet environnement pour provisionner l'infrastructure de l'instance de service. Les développeurs ne voient pas ce rôle et ne peuvent pas le modifier.
- Service avec pipeline : un modèle de service qui utilise le provisionnement AWS géré peut inclure une définition de pipeline écrite dans le schéma AWS CloudFormation YAML. AWS Proton crée également le pipeline en appelant AWS CloudFormation. Le rôle AWS Proton utilisé pour créer un pipeline est distinct de celui de chaque environnement individuel. Ce rôle est attribué AWS Proton séparément, une seule fois au niveau du AWS compte, et il est utilisé pour provisionner et gérer tous les AWS pipelines gérés. Ce rôle doit être autorisé à créer des pipelines et d'autres ressources dont vos pipelines ont besoin.

Les procédures suivantes montrent comment attribuer le rôle de pipeline AWS Proton.

AWS Proton console

Pour fournir le rôle de pipeline

1. Dans la [AWS Proton console](#), dans le volet de navigation, choisissez Paramètres > Paramètres du compte, puis sélectionnez Configurer.
2. Utilisez la section Rôle AWS géré par le pipeline pour configurer un rôle de pipeline nouveau ou existant pour le provisionnement AWS géré.

AWS Proton API

Pour fournir le rôle de pipeline

1. Utilisez l'action [UpdateAccountSettingsAPI](#).
2. Saisissez l'Amazon Resource Name (ARN) de votre rôle de pipeline `ServiceRoleArn` service.

AWS CLI

Pour fournir le rôle de pipeline

Exécutez la commande suivante :

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

Comment fonctionne CodeBuild le provisionnement

Lorsqu'un environnement ou un service utilise le CodeBuild provisionnement, l'infrastructure est provisionnée comme suit :

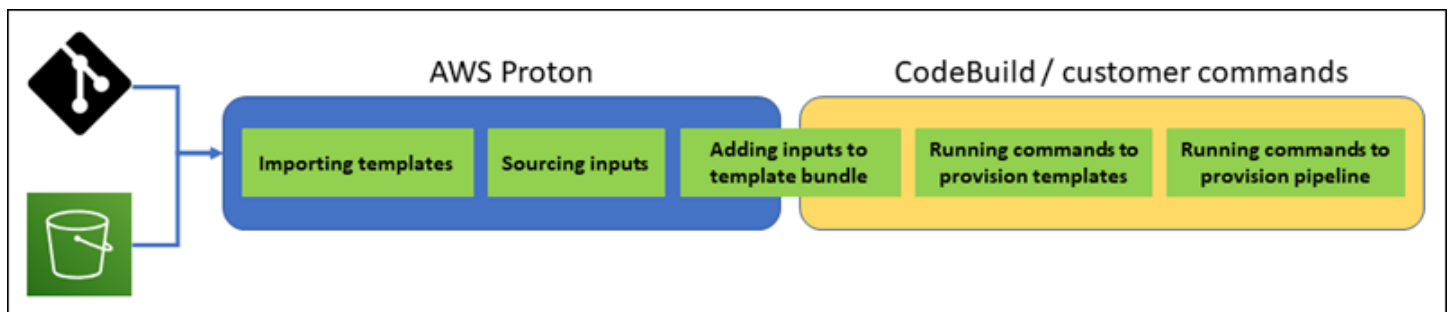
1. Un AWS Proton client (un administrateur ou un développeur) crée la AWS Proton ressource (un environnement ou un service). Le client sélectionne un modèle pour la ressource et fournit les paramètres requis. Pour plus d'informations, consultez la section suivante [the section called "Considérations relatives au CodeBuild provisionnement"](#).

2. AWS Proton restitue un fichier d'entrée avec les valeurs des paramètres d'entrée pour le provisionnement de la ressource.
3. AWS Proton appelle CodeBuild pour démarrer un travail. La tâche CodeBuild exécute les commandes du shell client spécifiées dans le modèle. Ces commandes fournissent l'infrastructure souhaitée, tout en lisant éventuellement les valeurs d'entrée.
4. Une fois le provisionnement terminé, la commande finale du client renvoie l'état du provisionnement à CodeBuild et appelle l'action de l'[NotifyResourceDeploymentStatusChange](#) AWS Proton API pour fournir des résultats, tels que l'identifiant Amazon VPC, le cas échéant.

⚠ Important

Assurez-vous que vos commandes renvoient correctement l'état du provisionnement CodeBuild et fournissent les sorties. Si ce n'est pas le cas, AWS Proton ne peut pas suivre correctement l'état du provisionnement et ne peut pas fournir de résultats corrects aux instances de service.

Le schéma suivant illustre les étapes qui s'exécutent dans AWS Proton et celles que vos commandes exécutent dans le cadre d'une tâche CodeBuild.



Considérations relatives au CodeBuild provisionnement

- Rôle de provisionnement de l'infrastructure : lorsqu'un environnement ou l'une des instances de service qui s'y exécutent peut utiliser le provisionnement CodeBuild basé, un administrateur doit configurer un rôle IAM (directement ou dans le cadre d'une connexion à un compte d'AWS Proton environnement). AWS Proton utilise ce rôle pour provisionner l'infrastructure de ces ressources de CodeBuild provisionnement. Le rôle doit disposer des autorisations nécessaires CodeBuild pour créer toutes les ressources que vos commandes dans les modèles de ces ressources fournissent.

Pour plus d'informations, consultez [the section called “Rôles IAM”](#) et [the section called “Exemples de politiques relatives aux rôles de service”](#).

- **Provisionnement de services** : lorsqu'un développeur déploie une instance de service qui utilise le CodeBuild provisionnement dans l'environnement, AWS Proton utilise le rôle attribué à cet environnement pour fournir l'infrastructure de l'instance de service. Les développeurs ne voient pas ce rôle et ne peuvent pas le modifier.
- **Service avec pipeline** : un modèle de service qui utilise le CodeBuild provisionnement peut inclure des commandes pour provisionner un pipeline. AWS Proton crée également le pipeline en appelant CodeBuild. Le rôle AWS Proton utilisé pour créer un pipeline est distinct de celui de chaque environnement individuel. Ce rôle est attribué AWS Proton séparément, une seule fois au niveau du AWS compte, et il est utilisé pour provisionner et gérer tous les pipelines CodeBuild basés. Ce rôle doit être autorisé à créer des pipelines et d'autres ressources dont vos pipelines ont besoin.

Les procédures suivantes montrent comment attribuer le rôle de pipeline AWS Proton.

AWS Proton console

Pour fournir le rôle de pipeline

1. Dans la [AWS Proton console](#), dans le volet de navigation, choisissez Paramètres > Paramètres du compte, puis sélectionnez Configurer.
2. Utilisez la section Rôle de provisionnement du pipeline Codebuild pour configurer un rôle de pipeline nouveau ou existant pour CodeBuild le provisionnement.

AWS Proton API

Pour fournir le rôle de pipeline

1. Utilisez l'action [UpdateAccountSettingsAPI](#).
2. Saisissez l'Amazon Resource Name (ARN) de votre rôle de pipelineCodebuildRoleArn service.

AWS CLI

Pour fournir le rôle de pipeline

Exécutez la commande suivante :

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

Comment fonctionne l'

Lorsqu'un environnement est configuré pour utiliser le provisionnement autogéré, l'infrastructure est provisionnée comme suit :

1. Un AWS Proton client (un administrateur ou un développeur) crée la AWS Proton ressource (un environnement ou un service). Le client sélectionne un modèle pour la ressource et fournit les paramètres requis. Pour un environnement, le client fournit également un référentiel d'infrastructure lié. Pour plus d'informations, consultez la section suivante [the section called "Considérations relatives à l'"](#).
2. AWS Proton génère un modèle Terraform complet. Il se compose d'un ou plusieurs fichiers Terraform, potentiellement dans plusieurs dossiers, et d'un fichier de `.tfvars` variables. AWS Proton écrit les valeurs des paramètres fournies lors de l'appel de création de ressources dans ce fichier de variables.
3. AWS Proton soumet un PR au référentiel d'infrastructure avec le modèle Terraform rendu.
4. Lorsque le client (administrateur ou développeur) fusionne le PR, l'automatisation du client déclenche le moteur de provisionnement pour démarrer le provisionnement de l'infrastructure à l'aide du modèle fusionné.

Note

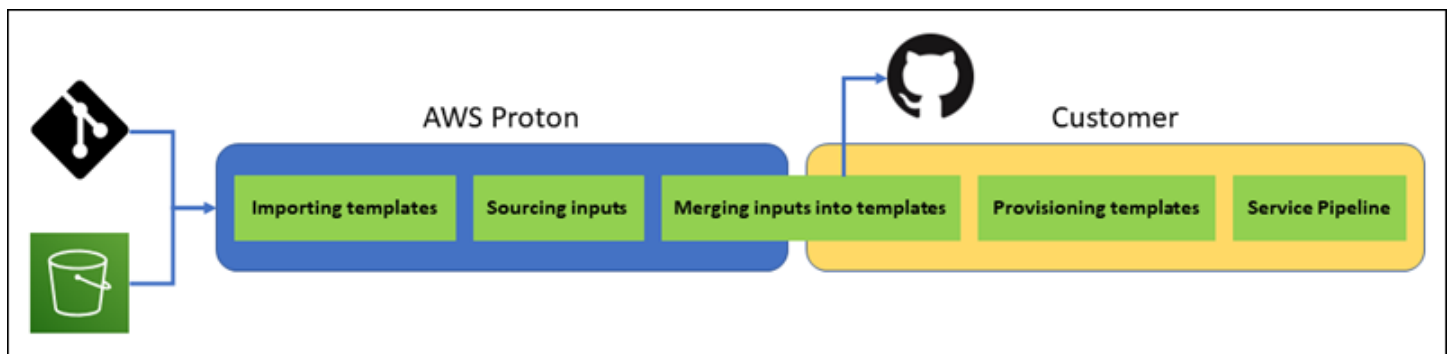
Si le client (administrateur ou développeur) ferme le PR, AWS Proton reconnaît le PR comme fermé et marque le déploiement comme annulé.

5. Lorsque le provisionnement est terminé, l'automatisation du client appelle l'action de [NotifyResourceDeploymentStatusChange](#) AWS Proton API pour indiquer qu'il est terminé, fournir le statut (réussite ou échec) et fournir des résultats, tels que l'identifiant Amazon VPC, le cas échéant.

⚠ Important

Assurez-vous que votre code d'automatisation rappelle l'état et AWS Proton les sorties du provisionnement. Si ce n'est pas le cas, vous pouvez considérer que le provisionnement est en attente plus longtemps qu'il ne le devrait et continuer à afficher le statut En cours.

Le schéma suivant illustre les étapes à AWS Proton suivre et celles effectuées par votre propre système de provisionnement.



Considérations relatives à l'

- **Référentiel d'infrastructure** : lorsqu'un administrateur configure un environnement pour le provisionnement autogéré, il doit fournir un référentiel d'infrastructure lié. AWS Proton soumet des PR à ce référentiel pour provisionner l'infrastructure de l'environnement et toutes les instances de service qui y sont déployées. L'action d'automatisation appartenant au client dans le référentiel doit assumer un rôle IAM doté des autorisations nécessaires pour créer toutes les ressources incluses dans vos modèles d'environnement et de service, ainsi qu'une identité qui reflète le compte de destination. AWS Pour un exemple d'GitHub action qui assume un rôle, consultez la section [Prise en charge d'un rôle](#) dans la documentation relative à l'action « Configurer les informations d'AWS identification » pour GitHub les actions.
- **Autorisations** : votre code de provisionnement doit s'authentifier auprès d'un compte si nécessaire (par exemple, vous authentifier auprès d'un AWS compte) et fournir une autorisation de provisionnement des ressources (par exemple, fournir un rôle).
- **Provisionnement de service** : lorsqu'un développeur déploie une instance de service qui utilise le provisionnement autogéré dans l'environnement, AWS Proton soumet un PR au référentiel associé

à l'environnement afin de fournir l'infrastructure de l'instance de service. Les développeurs ne voient pas le référentiel et ne peuvent pas le modifier.

Note

Les développeurs qui créent des services utilisent le même processus, quelle que soit la méthode de provisionnement, et la différence n'est pas prise en compte. Toutefois, avec le provisionnement autogéré, les développeurs peuvent être confrontés à un temps de réponse plus lent, car ils doivent attendre que quelqu'un (qui n'est peut-être pas eux-mêmes) fusionne le PR dans le référentiel d'infrastructure avant que le provisionnement ne puisse commencer.

- **Service avec pipeline** : un modèle de service pour un environnement doté d'un provisionnement autogéré peut inclure une définition de pipeline (par exemple, un AWS CodePipeline pipeline), écrite en Terraform HCL. Pour AWS Proton permettre le provisionnement de ces pipelines, un administrateur fournit un référentiel de pipelines lié à AWS Proton. Lors du provisionnement d'un pipeline, l'action d'automatisation appartenant au client dans le référentiel doit assumer un rôle IAM avec des autorisations pour approvisionner le pipeline et une identité qui reflète le compte de destination. AWS Le référentiel et le rôle du pipeline sont distincts de ceux utilisés pour chaque environnement individuel. Le référentiel lié est fourni AWS Proton séparément, une seule fois au niveau du AWS compte, et il est utilisé pour approvisionner et gérer tous les pipelines. Le rôle doit être autorisé à créer des pipelines et d'autres ressources dont vos pipelines ont besoin.

Les procédures suivantes montrent comment fournir le rôle et le pipeline AWS Proton.

AWS Proton console

Pour fournir le rôle de pipeline

1. Dans la [AWS Proton console](#), dans le volet de navigation, choisissez Paramètres > Paramètres du compte, puis sélectionnez Configurer.
2. Utilisez la section Référentiel du pipeline CI/CD pour configurer un lien de référentiel nouveau ou existant.

AWS Proton API

Pour fournir le rôle de pipeline

1. Utilisez l'action [UpdateAccountSettings](#) API.

- Indiquez le fournisseur, le nom et la branche de votre référentiel de pipeline dans le `pipelineProvisioningRepository` paramètre.

AWS CLI

Pour fournir le rôle de pipeline

Exécutez la commande suivante :

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

- Suppression des ressources provisionnées autogérées — Les modules Terraform peuvent inclure des éléments de configuration nécessaires au fonctionnement de Terraform, en plus des définitions de ressources. Par conséquent, AWS Proton impossible de supprimer tous les fichiers Terraform d'un environnement ou d'une instance de service. AWS Proton Marquez plutôt les fichiers à supprimer et mettez à jour un indicateur dans les métadonnées PR. Votre automatisation peut lire cet indicateur et l'utiliser pour déclencher une commande de destruction de terraform.

Terminologie AWS Proton

Modèle d'environnement

Définit l'infrastructure partagée, telle qu'un VPC ou un cluster, qui est utilisée par plusieurs applications ou ressources.

Ensemble de modèles d'environnement

Ensemble de fichiers que vous chargez pour créer et enregistrer un modèle d'environnement AWS Proton. Un bundle de modèles d'environnement contient les éléments suivants :

- Fichier de schéma qui définit l'infrastructure sous forme de paramètres d'entrée de code.
- Un fichier d'infrastructure sous forme de code (IaC) qui définit une infrastructure partagée, telle qu'un VPC ou un cluster, utilisée par plusieurs applications ou ressources.
- Un fichier manifeste qui répertorie le fichier IaC.

Environnement

Infrastructure partagée provisionnée, telle qu'un VPC ou un cluster, utilisée par plusieurs applications ou ressources.

Modèle de service

Définit le type d'infrastructure nécessaire au déploiement et à la maintenance d'une application ou d'un microservice dans un environnement.

Bundle

Ensemble de fichiers que vous importez pour créer et enregistrer un modèle de service AWS Proton. Un bundle de modèles de service contient les éléments suivants :

1. Fichier de schéma qui définit l'infrastructure en tant que code (IaC).
2. Fichier IaC qui définit l'infrastructure nécessaire au déploiement et à la maintenance d'une application ou d'un microservice dans un environnement.
3. Un fichier manifeste qui répertorie le fichier IaC.
4. Facultatif
 - a. Un fichier IaC qui définit l'infrastructure du pipeline de services.
 - b. Un fichier manifeste qui répertorie le fichier IaC.

Service

Infrastructure provisionnée nécessaire au déploiement et à la maintenance d'une application ou d'un microservice dans un environnement.

Instance de service

Infrastructure provisionnée qui prend en charge une application ou un microservice dans un environnement.

Pid' attente

Infrastructure provisionnée prenant en charge un pipeline.

Version du modèle

Version principale ou secondaire d'un modèle. Pour plus d'informations, veuillez consulter [Modèles versionnés](#).

Paramètres d'entrée

Défini dans un fichier de schéma et utilisé dans une infrastructure sous forme de fichier de code (IaC) afin que le fichier IaC puisse être utilisé de manière répétitive et pour divers cas d'utilisation.

Fichier de schéma

Définit l'infrastructure en tant que paramètres d'entrée du fichier de code.

Fichier de spécifications

Spécifie des valeurs pour l'infrastructure sous forme de paramètres d'entrée de fichier de code, tels que définis dans un fichier de schéma.

Fichier manifeste

Répertorie une infrastructure sous forme de fichier de code.

Création de modèles et création de packs pour AWS Proton

AWS Proton fournit des ressources pour vous sur la base de fichiers d'infrastructure sous forme de code (IaC). Vous décrivez l'infrastructure dans des fichiers IaC réutilisables. Pour que les fichiers soient réutilisables pour différents environnements et applications, vous les créez sous forme de modèles, vous définissez les paramètres d'entrée et vous utilisez ces paramètres dans les définitions IaC. Lorsque vous créez ultérieurement une ressource de provisionnement (environnement, instance de service ou composant), elle AWS Proton utilise un moteur de rendu qui combine les valeurs d'entrée avec un modèle pour créer un fichier IaC prêt à être provisionné.

Les administrateurs créent la plupart des modèles sous forme de lots de modèles, puis les téléchargent et les enregistrent dans AWS Proton. Le reste de cette page traite de ces ensembles AWS Proton de modèles. Les composants directement définis constituent une exception : les développeurs les créent et fournissent directement des fichiers modèles IaC. Pour plus d'informations sur les composants, consultez [Composants](#).

Rubriques

- [Packs de modèles](#)
- [AWS Proton paramètres](#)
- [AWS Proton infrastructure sous forme de fichiers de code](#)
- [Fichier de schéma](#)
- [Résumez les fichiers modèles pour AWS Proton](#)
- [Considérations relatives aux ensembles de modèles](#)

Packs de modèles

En tant qu'administrateur, vous [créez et enregistrez des modèles auprès](#) de AWS Proton. Vous utilisez ces modèles pour créer des environnements et des services. Lorsque vous créez un service, AWS Proton provisionne et déploie des instances de service dans des environnements sélectionnés. Pour de plus amples informations, veuillez consulter [AWS Proton pour les équipes de plateforme](#).

Pour créer et enregistrer un modèle AWS Proton, vous devez télécharger un ensemble de modèles contenant les fichiers d'infrastructure sous forme de code (IaC) AWS Proton nécessaires à la fourniture d'un environnement ou d'un service.

Un ensemble de modèles contient les éléments suivants :

- Un [fichier d'infrastructure en tant que code \(laC\)](#) avec un fichier [manifeste YAML répertoriant le fichier laC](#).
- Un [fichier de schéma YAML](#) pour les définitions des paramètres d'entrée de votre fichier laC.

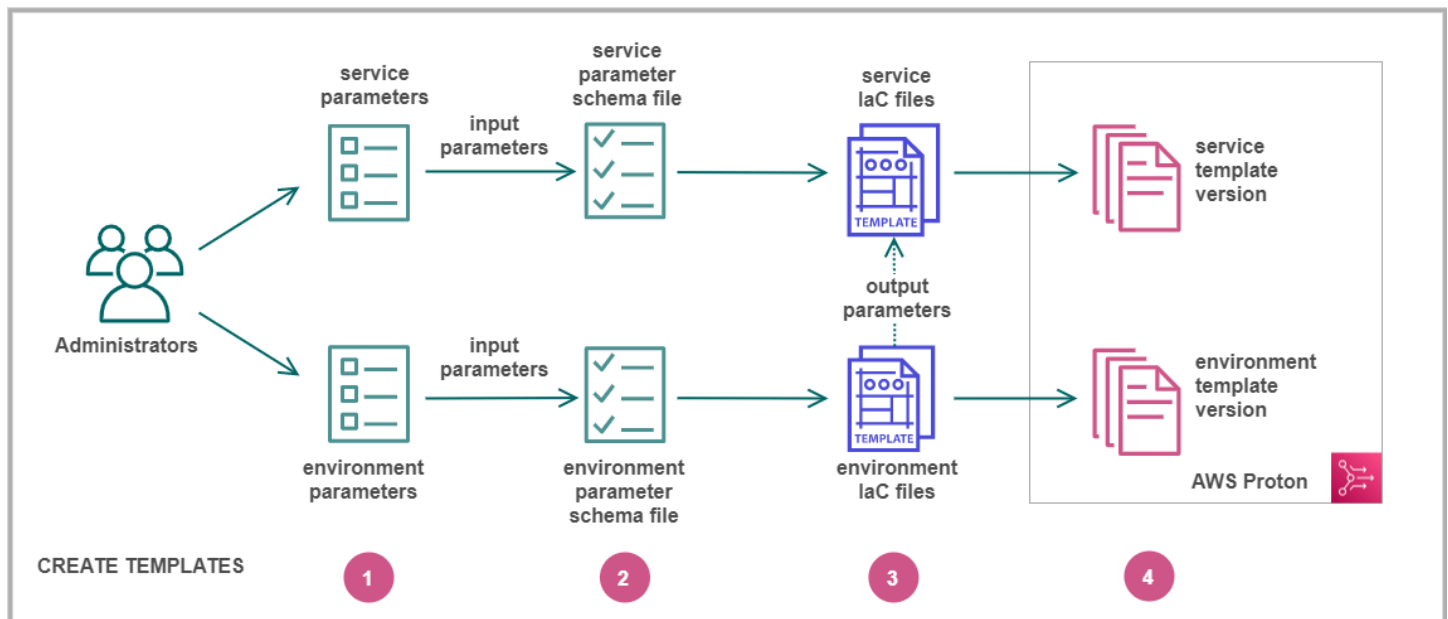
Un ensemble de modèles d' CloudFormation environnement contient un fichier laC.

Un ensemble CloudFormation de modèles de services contient un fichier laC pour les définitions d'instances de service et un autre fichier laC facultatif pour une définition de pipeline.

Les ensembles de modèles d'environnement et de service Terraform peuvent chacun contenir plusieurs fichiers laC.

AWS Proton nécessite un fichier de schéma de paramètres d'entrée. Lorsque vous créez vos fichiers laC, vous utilisez la syntaxe [Jinja](#) pour référencer vos paramètres d'entrée. AWS CloudFormation AWS Proton fournit des espaces de noms de paramètres que vous pouvez utiliser pour référencer [les paramètres](#) de vos fichiers laC.

Le schéma suivant montre un exemple d'étapes que vous pouvez suivre pour créer un modèle AWS Proton.



1

les [paramètres d'entrée](#).

Identifi

2

un [fichier de schéma](#) pour définir vos paramètres d'entrée.

Créez

3

des [fichiers IaC](#) qui font référence à vos paramètres d'entrée. Vous pouvez référencer les sorties des fichiers iAc de l'environnement en tant qu'entrées pour les fichiers iAc de votre service.

Créez

4

[une version de modèle](#) auprès de votre bundle de modèles AWS Proton et téléchargez-le.

Enregistrez

AWS Proton paramètres

Vous pouvez définir et utiliser les paramètres de votre infrastructure sous forme de fichiers de code (IaC) pour les rendre flexibles et réutilisables. Vous pouvez lire la valeur d'un paramètre dans vos fichiers IaC en vous référant au nom du paramètre dans l'espace de noms des AWS Proton paramètres. AWS Proton injecte des valeurs de paramètres dans les fichiers iAc rendus qu'il génère lors du provisionnement des ressources. Pour traiter les AWS CloudFormation paramètres IaC, AWS Proton utilise [Jinja](#). Pour traiter les paramètres Terraform IaC, AWS Proton génère un fichier de valeurs de paramètres Terraform et s'appuie sur la capacité de paramétrage intégrée à HCL.

Avec [CodeBuild provisionnement](#), AWS Proton génère un fichier d'entrée que votre code peut importer. Le fichier est un fichier JSON ou HCL, selon une propriété du manifeste de votre modèle. Pour de plus amples informations, veuillez consulter [the section called "CodeBuild paramètres de provisionnement"](#).

Vous pouvez faire référence aux paramètres de votre environnement, de vos services et de vos composants, aux fichiers iAc ou au code de provisionnement en respectant les exigences suivantes :

- La longueur de chaque nom de paramètre ne dépasse pas 100 caractères.
- La longueur de l'espace de noms des paramètres et du nom de ressource combinés ne dépasse pas la limite de caractères du nom de la ressource.

AWS Proton le provisionnement échoue si ces quotas sont dépassés.

Types de paramètres

Les types de paramètres suivants sont à votre disposition à titre de référence dans les fichiers AWS Proton IaC :

Paramètre d'entrée

Les environnements et les instances de service peuvent prendre des paramètres d'entrée que vous définissez dans un [fichier de schéma](#) que vous associez à l'environnement ou au modèle de service. Vous pouvez vous référer aux paramètres d'entrée d'une ressource dans le fichier IaC de la ressource. Les fichiers iAc du composant peuvent faire référence aux paramètres d'entrée de l'instance de service à laquelle le composant est attaché.

AWS Proton vérifie les noms des paramètres d'entrée par rapport à votre fichier de schéma et les associe aux paramètres référencés dans vos fichiers iAc pour injecter les valeurs d'entrée que vous fournissez dans un fichier de spécifications lors du provisionnement des ressources.

Paramètre de sortie

Vous pouvez définir des sorties dans n'importe lequel de vos fichiers iAc. Une sortie peut être, par exemple, le nom, l'ID ou l'ARN de l'une des ressources fournies par le modèle, ou elle peut être un moyen de passer par l'une des entrées du modèle. Vous pouvez faire référence à ces sorties dans les fichiers IaC d'autres ressources.

Dans les CloudFormation fichiers IaC, définissez les paramètres de sortie dans le `Outputs` : bloc. Dans un fichier Terraform iAc, définissez chaque paramètre de sortie à l'aide d'une `output` instruction.

Paramètre de ressource

AWS Proton crée automatiquement les paramètres AWS Proton des ressources. Ces paramètres exposent les propriétés de l'objet de AWS Proton ressource. Voici un exemple de paramètre de ressource `environment.name`.

Utilisation de AWS Proton paramètres dans vos fichiers iAc

Pour lire la valeur d'un paramètre dans un fichier IaC, vous devez vous référer au nom du paramètre dans l'espace de noms des AWS Proton paramètres. Pour les fichiers AWS CloudFormation iAc, vous devez utiliser la syntaxe Jinja et entourer le paramètre de paires d'accolades et de guillemets.

Le tableau suivant indique la syntaxe de référence pour chaque langage de modèle pris en charge, avec un exemple.

Langage du modèle	Syntaxe	Exemple : entrée d'environnement nommée « VPC »
CloudFormation	"{{ <i>parameter-name</i> }}"	"{{ environment.inputs.VPC }}"
Terraform	var. <i>parameter-name</i>	var.environment.inputs.VPC Définitions de variables Terraform générées

Note

Si vous utilisez des [paramètres CloudFormation dynamiques](#) dans votre fichier IaC, vous devez [y échapper pour éviter les](#) erreurs d'interprétation par Jinja. Pour plus d'informations, consultez [Résolution des problèmes de AWS Proton](#).

Le tableau suivant répertorie les noms d'espaces de noms pour tous les paramètres de AWS Proton ressources. Chaque type de fichier modèle peut utiliser un sous-ensemble différent de l'espace de noms des paramètres.

Fichier modèle	Type de paramètre	Nom du paramètre	Description
Environnement	ressource	environment. name	Nom de l'environnement
	input	environment.inputs. <i>input-name</i>	Entrées d'environnement définies par le schéma
Service	ressource	environment. name	Nom et Compte AWS ID de l'environnement
		environment. account_id	

Fichier modèle	Type de paramètre	Nom du paramètre	Description
	output	<code>environment.outputs. <i>output-name</i></code>	Sorties de fichiers iAC de l'environnement
	ressource	<code>service.branch_name</code> <code>service.name</code> <code>service.repository_connection_arn</code> <code>service.repository_id</code>	Nom du service et référentiel de code
	ressource	<code>service_instance.name</code>	Nom de l'instance de service
	input	<code>service_instance.inputs. <i>input-name</i></code>	Entrées d'instance de service définies par le schéma
	ressource	<code>service_instance.components.default.name</code>	Nom du composant par défaut joint
	output	<code>service_instance.components.default.outputs. <i>output-name</i></code>	Sorties du fichier iAC du composant par défaut joint
Pipeline	ressource	<code>service_instance.environment.name</code> <code>service_instance.environment.account_id</code>	Nom et Compte AWS ID de l'environnement de l'instance de service
	output	<code>service_instance.environment.outputs. <i>output-name</i></code>	Sorties de fichiers iAC de l'environnement d'instance de service

Fichier modèle	Type de paramètre	Nom du paramètre	Description
	input	pipeline.inputs. <i>input-name</i>	Entrées de pipeline définies par le schéma
	ressource	service.branch_name service.name service.repository_connection_arn service.repository_id	Nom du service et référentiel de code
	input	service_instance.inputs. <i>input-name</i>	Entrées d'instance de service définies par le schéma
	collection	{% for service_instance in service_instances %}...{% endfor %}	Ensemble d'instances de service que vous pouvez parcourir en boucle
	Composant	ressource	environment.name environment.account_id
output		environment.outputs. <i>output-name</i>	Sorties de fichiers iAC de l'environnement
ressource		service.branch_name service.name service.repository_connection_arn service.repository_id	Nom du service et référentiel de code (composants attachés)

Fichier modèle	Type de paramètre	Nom du paramètre	Description
	ressource	<code>service_instance.name</code>	Nom de l'instance de service (composants attachés)
	input	<code>service_instance.inputs.<i>input-name</i></code>	Entrées d'instance de service définies par le schéma (composants attachés)
	ressource	<code>component.name</code>	Nom du composant

Pour plus d'informations et des exemples, consultez les sous-rubriques relatives aux paramètres des fichiers modèles IaC pour différents types de ressources et différents langages de modèles.

Rubriques

- [Détails et exemples des paramètres du fichier CloudFormation IaC de l'environnement](#)
- [Détails et exemples des paramètres du fichier CloudFormation IaC du service](#)
- [Détails et exemples des paramètres du fichier CloudFormation iAC du composant](#)
- [Filtres de paramètres pour les fichiers CloudFormation IaC](#)
- [CodeBuild détails et exemples des paramètres de provisionnement](#)
- [Détails et exemples des paramètres du fichier d'infrastructure en tant que code \(IaC\) Terraform](#)

Détails et exemples des paramètres du fichier CloudFormation IaC de l'environnement

Vous pouvez définir et référencer les paramètres de votre infrastructure d'environnement sous forme de fichiers de code (IaC). Pour une description détaillée des AWS Proton paramètres, des types de paramètres, de l'espace de noms des paramètres et de la façon d'utiliser les paramètres dans vos fichiers iAc, consultez [the section called "Paramètres"](#).

Définir les paramètres de l'environnement

Vous pouvez définir des paramètres d'entrée et de sortie pour les fichiers iAc d'environnement.

- Paramètres d'entrée : définissez les paramètres d'entrée de l'environnement dans votre [fichier de schéma](#).

La liste suivante inclut des exemples de paramètres d'entrée d'environnement pour des cas d'utilisation typiques.

- Valeurs CIDR VPC
- Paramètres de l'équilibreur de charge
- Paramètres de base de données
- Un délai d'expiration pour le bilan de santé

En tant qu'administrateur, vous pouvez fournir des valeurs pour les paramètres d'entrée lorsque vous [créez un environnement](#) :

- Utilisez la console pour remplir un formulaire basé sur un schéma que AWS Proton fournit.
- Utilisez la CLI pour fournir une spécification qui inclut les valeurs.
- Paramètres de sortie — Définissez les sorties d'environnement dans les fichiers iAc de votre environnement. Vous pouvez ensuite faire référence à ces sorties dans les fichiers IaC d'autres ressources.

Lire les valeurs des paramètres dans les fichiers iAc de l'environnement

Vous pouvez lire les paramètres relatifs à l'environnement dans les fichiers iAc de l'environnement. Vous pouvez lire la valeur d'un paramètre en faisant référence au nom du paramètre dans l'espace de noms des AWS Proton paramètres.

- Paramètres d'entrée — Lisez la valeur d'entrée d'un environnement en la référençant `environment.inputs.input-name`.
- Paramètres des ressources : lisez les paramètres AWS Proton des ressources en faisant référence à des noms tels que `environment.name`.

Note

Aucun paramètre de sortie d'autres ressources n'est disponible pour les fichiers IaC de l'environnement.

Exemples de fichiers IaC d'environnement et de service avec paramètres

L'exemple suivant illustre la définition et la référence de paramètres dans un fichier IaC d'environnement. L'exemple montre ensuite comment les paramètres de sortie d'environnement définis dans le fichier IaC d'environnement peuvent être référencés dans un fichier IaC de service.

Exemple Fichier CloudFormation IaC d'environnement

Notez ce qui suit dans cet exemple :

- L'espace de `environment.inputs` noms fait référence aux paramètres d'entrée de l'environnement.
- Le `StoreInputValue` paramètre Amazon EC2 Systems Manager (SSM) concatène les entrées de l'environnement.
- La `MyEnvParameterValue` sortie expose la même concaténation de paramètres d'entrée qu'un paramètre de sortie. Trois paramètres de sortie supplémentaires exposent également les paramètres d'entrée individuellement.
- Six paramètres de sortie supplémentaires exposent les ressources fournies par l'environnement.

```
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

```

Exemple Fichier CloudFormation IaC du service

L'espace de `environment.outputs` noms fait référence aux sorties d'environnement d'un fichier IaC d'environnement. Par exemple, le nom `environment.outputs.ClusterName` indique la valeur du paramètre de sortie de `ClusterName` l'environnement.

```

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'

[...]
```

Détails et exemples des paramètres du fichier CloudFormation IaC du service

Vous pouvez définir et référencer les paramètres de votre infrastructure de service et de pipeline sous forme de fichiers de code (iAc). Pour une description détaillée des AWS Proton paramètres, des types de paramètres, de l'espace de noms des paramètres et de la façon d'utiliser les paramètres dans vos fichiers iAc, consultez [the section called "Paramètres"](#).

Définir les paramètres du service

Vous pouvez définir des paramètres d'entrée et de sortie pour les fichiers IaC du service.

- Paramètres d'entrée : définissez les paramètres d'entrée de l'instance de service dans votre [fichier de schéma](#).

La liste suivante inclut des exemples de paramètres d'entrée de service pour des cas d'utilisation typiques.

- Port
- Taille de la tâche
- Image
- Nombre souhaité
- fichier Docker
- Commande de test unitaire

Vous fournissez des valeurs pour les paramètres d'entrée lorsque vous [créez un service](#) :

- Utilisez la console pour remplir un formulaire basé sur un schéma qui AWS Proton fournit.
- Utilisez la CLI pour fournir une spécification qui inclut les valeurs.
- Paramètres de sortie — Définissez les sorties des instances de service dans vos fichiers IaC de service. Vous pouvez ensuite faire référence à ces sorties dans les fichiers IaC d'autres ressources.

Lire les valeurs des paramètres dans les fichiers IaC du service

Vous pouvez lire les paramètres relatifs au service et à d'autres ressources dans les fichiers IaC du service. Vous pouvez lire la valeur d'un paramètre en faisant référence au nom du paramètre dans l'espace de noms des AWS Proton paramètres.

- Paramètres d'entrée — Lisez la valeur d'entrée d'une instance de service en la référençant `service_instance.inputs.input-name`.
- Paramètres des ressources : lisez les paramètres AWS Proton des ressources en faisant référence à des noms tels que `service.nameservice_instance.name`, et `environment.name`.
- Paramètres de sortie — Lisez les sorties d'autres ressources en faisant référence à `environment.outputs.output-name` ou `service_instance.components.default.outputs.output-name`.

Exemple de fichier IaC de service avec paramètres

L'exemple suivant est un extrait d'un fichier IaC de service CloudFormation . L'espace de `environment.outputs` noms fait référence aux sorties du fichier IaC de l'environnement. L'espace de `service_instance.inputs` noms fait référence aux paramètres d'entrée de l'instance de service. La `service_instance.name` propriété fait référence à un paramètre de AWS Proton ressource.

```
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

```

Détails et exemples des paramètres du fichier CloudFormation iAc du composant

Vous pouvez définir et référencer les paramètres de votre infrastructure de composants sous forme de fichiers de code (IaC). Pour une description détaillée des AWS Proton paramètres, des types de paramètres, de l'espace de noms des paramètres et de la façon d'utiliser les paramètres dans vos fichiers iAc, consultez [the section called "Paramètres"](#). Pour plus d'informations sur les composants, consultez [Composants](#).

Définir les paramètres de sortie des composants

Vous pouvez définir les paramètres de sortie dans les fichiers iAc de vos composants. Vous pouvez ensuite faire référence à ces sorties dans les fichiers IaC du service.

Note

Vous ne pouvez pas définir d'entrées pour les fichiers iAc des composants. Les composants attachés peuvent obtenir des entrées de l'instance de service à laquelle ils sont attachés. Les composants détachés n'ont pas d'entrées.

Lire les valeurs des paramètres dans les fichiers iAc des composants

Vous pouvez lire les paramètres relatifs au composant et à d'autres ressources dans les fichiers iAc du composant. Vous pouvez lire la valeur d'un paramètre en faisant référence au nom du paramètre dans l'espace de noms des AWS Proton paramètres.

- Paramètres d'entrée — Lisez la valeur d'entrée d'une instance de service attachée en la référençant `service_instance.inputs.input-name`.

- Paramètres des ressources : lisez les paramètres AWS Proton des ressources en faisant référence à des noms tels que `component.nameservice.name`, `service_instance.name`, `environment.name`.
- Paramètres de sortie — Lisez les sorties de l'environnement en les référençant `environment.outputs.output-name`.

Exemples de fichiers IaC de composants et de services avec paramètres

L'exemple suivant montre un composant qui provisionne un bucket Amazon Simple Storage Service (Amazon S3) et la politique d'accès associée, et qui expose les Amazon Resource Names (ARN) des deux ressources sous forme de sorties de composants. Un modèle de service IaC ajoute les sorties des composants en tant que variables d'environnement de conteneur d'une tâche Amazon Elastic Container Service (Amazon ECS) afin de rendre les sorties disponibles pour le code exécuté dans le conteneur, et ajoute la politique d'accès au bucket au rôle de la tâche. Le nom du compartiment est basé sur les noms de l'environnement, du service, de l'instance de service et du composant, ce qui signifie que le compartiment est couplé à une instance spécifique du modèle de composant étendant une instance de service spécifique. Les développeurs peuvent créer plusieurs composants personnalisés sur la base de ce modèle de composants, afin de fournir des compartiments Amazon S3 pour différentes instances de service et différents besoins fonctionnels.

L'exemple montre comment vous utilisez la `{{ ... }}` syntaxe Jinja pour faire référence aux paramètres des composants et autres ressources dans votre fichier IaC de service. Vous pouvez utiliser `{% if ... %}` des instructions pour ajouter des blocs d'instructions uniquement lorsqu'un composant est attaché à l'instance de service. Les `proton_cfn_*` mots clés sont des filtres que vous pouvez utiliser pour nettoyer et formater les valeurs des paramètres de sortie. Pour plus d'informations sur les filtres, consultez [the section called "CloudFormation filtres de paramètres"](#).

En tant qu'administrateur, vous créez le fichier modèle du service iAC.

Exemple fichier CloudFormation IaC de service utilisant un composant

```
# 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:
    # ...

```

En tant que développeur, vous créez le fichier modèle du composant iAc.

Exemple fichier CloudFormation iAc du composant

```

# 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

```

Lors du AWS Proton rendu d'un AWS CloudFormation modèle pour votre instance de service et du remplacement de tous les paramètres par des valeurs réelles, le modèle peut ressembler au fichier suivant.

Exemple fichier IaC CloudFormation rendu par une instance de service

```

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:
    # ...
```

Filtres de paramètres pour les fichiers CloudFormation IaC

Lorsque vous faites référence à des [AWS Proton paramètres](#) dans vos fichiers AWS CloudFormation IaC, vous pouvez utiliser des modificateurs Jinja appelés filtres pour valider, filtrer et formater les valeurs des paramètres avant qu'elles ne soient insérées dans le modèle de rendu. [Les validations de filtres sont particulièrement utiles lorsqu'il s'agit de faire référence aux paramètres de sortie des composants, car la création et l'attachement des composants sont effectués par les développeurs, et un administrateur utilisant les sorties des composants dans un modèle d'instance de service peut souhaiter vérifier leur existence et leur validité.](#) Cependant, vous pouvez utiliser des filtres dans n'importe quel fichier Jinja IaC.

Les sections suivantes décrivent et définissent les filtres de paramètres disponibles, et fournissent des exemples. AWS Proton définit la plupart de ces filtres. Le default filtre est un filtre intégré à Jinja.

Formater les propriétés de l'environnement pour les tâches Amazon ECS

Déclaration

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

Description

Ce filtre met en forme une liste de sorties à utiliser dans une [propriété d'environnement](#) dans la ContainerDefinition section d'une définition de tâche Amazon Elastic Container Service (Amazon ECS).

Définissez raw sur False pour valider également la valeur du paramètre. Dans ce cas, la valeur doit correspondre à l'expression régulière `^[a-zA-Z0-9_-]*$`. Si la valeur échoue à cette validation, le rendu du modèle échoue.

Exemple

Avec le modèle de composant personnalisé suivant :

```

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

```

Et le modèle de service suivant :

```

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

```

Le modèle de service rendu est le suivant :

```

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

```

Propriétés de l'environnement de formatage pour les fonctions Lambda

Déclaration

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

Description

Ce filtre met en forme une liste de sorties à utiliser dans une [propriété d'environnement](#) dans la `Properties` section de définition d'une AWS Lambda fonction.

Définissez `raw` sur `False` pour valider également la valeur du paramètre. Dans ce cas, la valeur doit correspondre à l'expression régulière `^[a-zA-Z0-9_-]*$`. Si la valeur échoue à cette validation, le rendu du modèle échoue.

Exemple

Avec le modèle de composant personnalisé suivant :

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

Et le modèle de service suivant :

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

Le modèle de service rendu est le suivant :

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

Extraire les ARN de la politique IAM à inclure dans les rôles IAM

Déclaration

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

Description

Ce filtre met en forme une liste de sorties à utiliser dans une [ManagedPolicyArns propriété](#) dans la `Properties` section de définition d'un rôle AWS Identity and Access Management (IAM). Le filtre utilise l'expression régulière `^arn:[a-zA-Z-]+:iam::\d{12}:policy/` pour extraire les ARN de politique IAM valides de la liste des paramètres de sortie. Vous pouvez utiliser ce filtre pour ajouter des politiques dans les valeurs des paramètres de sortie à une définition de rôle IAM dans un modèle de service.

Exemple

Avec le modèle de composant personnalisé suivant :

```
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

```

Et le modèle de service suivant :

```

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:
    # ...

```

Le modèle de service rendu est le suivant :

```

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:
    # ...

```

Nettoyez les valeurs des propriétés

Déclaration

```
string # proton_cfn_sanitize # string
```

Description

Il s'agit d'un filtre à usage général. Utilisez-le pour valider la sécurité d'une valeur de paramètre. Le filtre vérifie que la valeur correspond à l'expression régulière `^[a-zA-Z0-9_-]*$` ou qu'il s'agit d'un Amazon Resource Name (ARN) valide. Si la valeur échoue à cette validation, le rendu du modèle échoue.

Exemple

Avec le modèle de composant personnalisé suivant :

```

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*

- La référence suivante dans un modèle de service :

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

S'affiche comme suit :

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

- La référence suivante dans un modèle de service :

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

Résultats avec l'erreur de rendu suivante :

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

- La référence suivante dans un modèle de service :

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

S'affiche comme suit :

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

Fournir des valeurs par défaut pour les références inexistantes

Description

Le `default` filtre fournit une valeur par défaut lorsqu'il n'existe aucune référence à un espace de noms. Utilisez-le pour écrire des modèles robustes qui peuvent être rendus sans échec même lorsque le paramètre auquel vous faites référence est manquant.

Exemple

La référence suivante dans un modèle de service entraîne l'échec du rendu du modèle si l'instance de service n'a pas de composant directement défini (par défaut) attaché, ou si le composant attaché n'a pas de sortie nommée `test`.

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

Pour éviter ce problème, ajoutez le `default` filtre.

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

CodeBuild détails et exemples des paramètres de provisionnement

Vous pouvez définir des paramètres dans vos modèles pour les AWS Proton ressources CodeBuild basées et référencer ces paramètres dans votre code de provisionnement. Pour une description détaillée des AWS Proton paramètres, des types de paramètres, de l'espace de noms des paramètres et de la façon d'utiliser les paramètres dans vos fichiers `iAc`, consultez [the section called "Paramètres"](#).

Note

Vous pouvez utiliser le CodeBuild provisionnement avec des environnements et des services. Pour le moment, vous ne pouvez pas approvisionner les composants de cette façon.

Paramètres d'entrée

Lorsque vous créez une AWS Proton ressource, comme un environnement ou un service, vous fournissez des valeurs pour les paramètres d'entrée qui sont définis dans le [fichier de schéma](#) de votre modèle. Lorsque la ressource que vous créez utilise [CodeBuild approvisionnement](#), AWS Proton affiche ces valeurs d'entrée dans un fichier d'entrée. Votre code d'approvisionnement peut importer et obtenir des valeurs de paramètres à partir de ce fichier.

Pour un exemple de CodeBuild modèle, voir [the section called “CodeBuild offre groupée”](#). Pour plus d'informations sur les fichiers manifeste, consultez [the section called “Manifeste et résumé”](#).

L'exemple suivant est un fichier d'entrée JSON généré lors du provisionnement CodeBuild basé sur une instance de service.

Exemple : utilisation du AWS CDK with CodeBuild provisioning

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

Paramètres de sortie

Pour renvoyer les résultats de mise à disposition des ressources AWS Proton, votre code de provisionnement peut générer un fichier JSON nommé `proton-outputs.json` avec les valeurs des paramètres de sortie définis dans le fichier de [schéma](#) de votre modèle. Par exemple, l'option `--outputs-file` argument de la `cdk deploy` commande indique de générer un fichier JSON avec des sorties de provisionnement. AWS CDK Si votre ressource utilise le AWS CDK, spécifiez la commande suivante dans votre CodeBuild modèle de manifeste :

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

AWS Proton recherche ce fichier JSON. Si le fichier existe une fois que votre code d'approvisionnement a été correctement terminé, il AWS Proton lit les valeurs des paramètres de sortie à partir de celui-ci.

Détails et exemples des paramètres du fichier d'infrastructure en tant que code (IaC) Terraform

Vous pouvez inclure des variables d'entrée Terraform dans les `variable.tf` fichiers de votre ensemble de modèles. Vous pouvez également créer un schéma pour créer des variables AWS Proton gérées. AWS Proton crée une variable `.tf` files à partir de votre fichier de schéma. Pour de plus amples informations, veuillez consulter [the section called "Fichiers Terraform IaC"](#).

Pour référencer les AWS Proton variables définies par votre schéma dans votre `infrastructure.tf` files, vous utilisez les AWS Proton espaces de noms indiqués dans la table Paramètres et espaces de noms pour Terraform IaC. Par exemple, vous pouvez utiliser `var.environment.inputs.vpc_cidr`. Entre guillemets, entourez ces variables de crochets simples et ajoutez un signe dollar devant la première accolade (par exemple, `"${var.environment.inputs.vpc_cidr}"`).

L'exemple suivant montre comment utiliser les espaces de noms pour inclure des AWS Proton paramètres dans un environnement. `.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 infrastructure sous forme de fichiers de code

Les principaux éléments du bundle de modèles sont des fichiers d'infrastructure sous forme de code (IaC) qui définissent les ressources et les propriétés de l'infrastructure que vous souhaitez provisionner. AWS CloudFormation et d'autres infrastructures, car les moteurs de code utilisent ces types de fichiers pour provisionner les ressources de l'infrastructure.

Note

Un fichier IaC peut également être utilisé indépendamment des ensembles de modèles, en tant qu'entrée directe vers des composants directement définis. Pour plus d'informations sur les composants, consultez [Composants](#).

AWS Proton supporte actuellement deux types de fichiers IaC :

- [CloudFormation](#) files — Utilisé pour le provisionnement AWS géré. AWS Proton utilise Jinja en plus du format de fichier CloudFormation modèle pour le paramétrage.
- Fichiers [Terraform HCL](#) : utilisés pour le provisionnement autogéré. HCL prend en charge le paramétrage de manière native.

Vous ne pouvez pas provisionner AWS Proton des ressources à l'aide d'une combinaison de plusieurs méthodes de provisionnement. Vous devez utiliser l'un ou l'autre. Vous ne pouvez pas déployer un service de provisionnement AWS géré dans un environnement de provisionnement autogéré, ou vice versa.

Pour plus d'informations, consultez [the section called “Méthodes de provisionnement”](#), [Environnements](#), [Services](#) et [Composants](#).

AWS CloudFormation fichiers IaC

Découvrez comment utiliser l' AWS CloudFormation infrastructure sous forme de fichiers de code avec AWS Proton. AWS CloudFormation est un service d'infrastructure sous forme de code (IaC) qui vous aide à modéliser et à configurer vos AWS ressources. Vous définissez les ressources de votre infrastructure dans des modèles, en utilisant Jinja en plus du format de fichier CloudFormation modèle pour le paramétrage. AWS Proton développe les paramètres et affiche le CloudFormation modèle complet. CloudFormation provisionne les ressources définies sous forme de CloudFormation pile. Pour plus d'informations, consultez la section [Contenu AWS CloudFormation](#) du guide de AWS CloudFormation l'utilisateur.

AWS Proton prend en charge [AWS le provisionnement géré](#) pour CloudFormation IaC.

Commencez avec votre propre infrastructure existante sous forme de fichiers de code

Vous pouvez adapter votre propre infrastructure existante sous forme de fichiers de code (IaC) pour les utiliser avec AWS Proton.

Les AWS CloudFormation exemples suivants, l'[exemple 1](#) et l'[exemple 2](#), représentent vos propres fichiers CloudFormation IaC existants. CloudFormation peut utiliser ces fichiers pour créer deux CloudFormation piles différentes.

Dans l'[exemple 1](#), le fichier CloudFormation IaC est configuré pour provisionner l'infrastructure à partager entre les applications de conteneur. Dans cet exemple, des paramètres d'entrée sont ajoutés afin que vous puissiez utiliser le même fichier IaC pour créer plusieurs ensembles d'infrastructures provisionnées. Chaque ensemble peut avoir des noms différents ainsi qu'un ensemble différent de valeurs de VPC et de CIDR de sous-réseau. En tant qu'administrateur ou développeur, vous fournissez des valeurs pour ces paramètres lorsque vous utilisez un fichier IaC pour provisionner des ressources d'infrastructure CloudFormation. Pour votre commodité, ces paramètres d'entrée sont marqués de commentaires et référencés à plusieurs reprises dans l'exemple. Les sorties sont définies à la fin du modèle. Ils peuvent être référencés dans d'autres fichiers CloudFormation IaC.

Dans l'[exemple 2](#), le fichier CloudFormation iAC est configuré pour déployer une application sur l'infrastructure provisionnée à partir de l'exemple 1. Les paramètres sont commentés pour votre commodité.

Exemple 1 : fichier CloudFormation IaC

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

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

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

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

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

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

```

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

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

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

```

```

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

```

Exemple 2 : fichier CloudFormation IaC

```

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

```



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

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

```

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

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

    LogConfiguration:
      LogDriver: 'awslogs'
      Options:

```

```

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

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

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

```

```

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

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

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

```

```

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

```

```

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

```

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

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

```

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

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

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

```

```

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

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

```

Vous pouvez adapter ces fichiers pour les utiliser avec AWS Proton.

Transférez votre infrastructure sous forme de code à AWS Proton

Avec de légères modifications, vous pouvez utiliser l'[exemple 1](#) en tant que fichier d'infrastructure en tant que code (IaC) pour un ensemble de modèles d'environnement AWS Proton utilisé pour déployer un environnement (comme indiqué dans l'[exemple 3](#)).

Au lieu d'utiliser les CloudFormation paramètres, vous utilisez la syntaxe [Jinja](#) pour référencer les paramètres que vous avez définis dans un [fichier de schéma](#) basé sur une [API ouverte](#). Ces paramètres d'entrée sont commentés pour votre commodité et référencés plusieurs fois dans le fichier IaC. De cette façon, AWS Proton vous pouvez auditer et vérifier les valeurs des paramètres. Il peut également faire correspondre et insérer les valeurs des paramètres de sortie d'un fichier IaC aux paramètres d'un autre fichier IaC.

En tant qu'administrateur, vous pouvez ajouter l'espace de AWS Proton `environment.inputs.noms` aux paramètres d'entrée. Lorsque vous référencez les sorties d'un fichier IaC de l'environnement dans un fichier IaC de service, vous pouvez ajouter l'espace de noms aux sorties (par exemple, `environment.outputs.ClusterName`). Enfin, vous les entourez d'accolades et de guillemets.

Avec ces modifications, vos fichiers CloudFormation IaC peuvent être utilisés par AWS Proton.

Exemple 3 : infrastructure d' AWS Proton environnement sous forme de fichier de code

```
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::VPCGatewayAttachement
  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'

```

Les fichiers IaC de l'[exemple 1](#) et de l'[exemple 3](#) produisent des CloudFormation piles légèrement différentes. Les paramètres sont affichés différemment dans les fichiers modèles de pile. Le fichier de modèle de CloudFormation pile Example 1 affiche les étiquettes de paramètres (clés) dans la vue du modèle de pile. Le fichier modèle de pile AWS Proton CloudFormation d'infrastructure de l'exemple 3 affiche les valeurs des paramètres. AWS Proton les paramètres d'entrée n'apparaissent pas dans la vue des paramètres de la CloudFormation pile de consoles.

Dans l'[exemple 4](#), le fichier AWS Proton IaC du service correspond à l'[exemple 2](#).

Exemple 4 : fichier IaC de l'instance de AWS Proton service

```

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}"

```

Dans l'[exemple 5](#), le fichier iAC du AWS Proton pipeline approvisionne l'infrastructure du pipeline pour prendre en charge les instances de service fournies par l'[exemple 4](#).

Exemple 5 : fichier AWS Proton IaC du pipeline de services

```

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 ensemble de modèles de provisionnement

Avec le CodeBuild provisionnement, au lieu d'utiliser des modèles IaC pour afficher les fichiers IaC et les exécuter à l'aide d'un moteur de provisionnement IaC, AWS Proton il suffit d'exécuter vos commandes shell. Pour ce faire, AWS Proton créez un AWS CodeBuild projet pour l'environnement, dans le compte d'environnement, et lancez une tâche pour exécuter vos commandes pour chaque création ou mise à jour de AWS Proton ressource. Lorsque vous créez un ensemble de modèles, vous fournissez un manifeste qui spécifie les commandes de provisionnement et de déprovisionnement de l'infrastructure, ainsi que tous les programmes, scripts et autres fichiers dont ces commandes peuvent avoir besoin. Vos commandes peuvent lire les entrées que AWS Proton fournissent et sont responsables du provisionnement ou du déprovisionnement de l'infrastructure et de la génération de valeurs de sortie.

Le manifeste indique également comment AWS Proton doit être affiché le fichier d'entrée que votre code peut saisir et à partir duquel les valeurs d'entrée doivent être obtenues. Il peut être rendu au format JSON ou HCL. Pour plus d'informations sur les paramètres d'entrée, consultez [the section called "CodeBuild paramètres de provisionnement"](#). Pour plus d'informations sur les fichiers manifeste, consultez [the section called "Manifeste et résumé"](#).

Note

Vous pouvez utiliser le CodeBuild provisionnement avec des environnements et des services. Pour le moment, vous ne pouvez pas approvisionner les composants de cette façon.

Exemple : utilisation du AWS CDK with CodeBuild provisioning

À titre d'exemple d'utilisation du CodeBuild provisionnement, vous pouvez inclure du code qui utilise les AWS ressources AWS Cloud Development Kit (AWS CDK) pour provisionner (déployer) et déprovisionner (détruire), ainsi qu'un manifeste qui installe le CDK et exécute votre code CDK.

Les sections suivantes répertorient des exemples de fichiers que vous pouvez inclure dans un ensemble de modèles de CodeBuild provisionnement qui approvisionne un environnement à l'aide du AWS CDK.

Manifeste

Le fichier manifeste suivant spécifie le CodeBuild provisionnement et inclut les commandes nécessaires pour installer et utiliser le AWS CDK, le traitement des fichiers de sortie et les rapports sur les résultats vers AWS Proton.

Exemple 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

Le fichier de schéma suivant définit les paramètres de l'environnement. Votre AWS CDK code peut faire référence aux valeurs de ces paramètres lors du déploiement.

Exemple schéma/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 fichiers

Les fichiers suivants constituent un exemple de projet CDK Node.js.

Exemple infrastructure/package.json

```
{
  "name": "ProtonEnvironment",
  "version": "0.1.0",
  "bin": {
    "ProtonEnvironment": "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 infrastructure/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 infrastructure/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/.ts ProtonEnvironment

```
#!/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/ .ts ProtonEnvironmentStack

```
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`
    });
  }
}
```

Fichier d'entrée rendu

Lorsque vous créez un environnement à l'aide d'un modèle de provisionnement CodeBuild basé, AWS Proton affiche un fichier d'entrée avec les [valeurs de paramètres d'entrée](#) que vous avez fournies. Votre code peut faire référence à ces valeurs. Le fichier suivant est un exemple de fichier d'entrée rendu.

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"
  }
}
}
```

Fichiers Terraform iAC

Apprenez à utiliser l'infrastructure Terraform sous forme de fichiers de code (IaC) avec AWS Proton. [Terraform](#) est un moteur IaC open source largement utilisé qui a été développé par [HashiCorp](#). Les modules Terraform sont développés dans le langage HashiCorp HCL et prennent en charge plusieurs fournisseurs d'infrastructures dorsales, dont Amazon Web Services.

AWS Proton prend en charge le [provisionnement autogéré](#) pour Terraform iAC.

Pour un exemple complet de référentiel de provisionnement qui répond aux pull requests et implémente le provisionnement de l'infrastructure, voir le modèle d'[automatisation Terraform OpenSource GitHub Actions](#) pour on. AWS Proton GitHub

Comment fonctionne le provisionnement autogéré avec les fichiers groupés de modèles Terraform iAc :

1. Lorsque vous [créez un environnement](#) à partir de ensembles de modèles Terraform, AWS Proton compile vos `.tf` fichiers avec des paramètres de console ou d'entrée. `spec file`
2. Il effectue une pull request pour fusionner les fichiers iAc compilés [avec le référentiel auprès duquel vous vous êtes enregistré AWS Proton](#).
3. Si la demande est approuvée, AWS Proton attend le statut d'approvisionnement que vous fournissez.
4. Si la demande est rejetée, la création de l'environnement est annulée.
5. Si le délai d'expiration de la pull request est dépassé, la création de l'environnement n'est pas terminée.

AWS Proton avec les considérations relatives à Terraform IaC :

- AWS Proton ne gère pas votre approvisionnement en Terraform.
- Vous devez [enregistrer un référentiel de provisioning](#) auprès AWS Proton de. AWS Proton effectue des pull requests sur ce dépôt.
- Vous devez [créer une CodeStar connexion](#) pour vous connecter AWS Proton à votre référentiel de provisioning.

- Pour effectuer un provisionnement à partir de fichiers IaC AWS Proton compilés, vous devez répondre aux AWS Proton pull requests. AWS Proton effectue des pull requests après la création et la mise à jour des actions de l'environnement et du service. Pour plus d'informations, consultez [Environnements AWS Proton](#) et [AWS Proton services](#).
- Pour approvisionner un pipeline à partir de fichiers iAc AWS Proton compilés, vous devez [créer un référentiel de pipeline CI/CD](#).
- Votre automatisation du provisionnement basée sur les pull requests doit inclure des étapes pour signaler tout AWS Proton changement d'état des AWS Proton ressources provisionnées. Vous pouvez utiliser l' AWS Proton [NotifyResourceDeploymentStatusChange API](#).
- Vous ne pouvez pas déployer de services, de pipelines et de composants créés à partir de fichiers CloudFormation iAC dans des environnements créés à partir de fichiers Terraform iAC.
- Vous ne pouvez pas déployer de services, de pipelines et de composants créés à partir de fichiers IaC Terraform dans des environnements créés à partir de fichiers CloudFormation iAC.

Lorsque vous préparez vos fichiers Terraform iAC pour AWS Proton, vous attachez des espaces de noms à vos variables d'entrée, comme indiqué dans les exemples suivants. Pour plus d'informations, consultez [Parameters](#).

Exemple 1 : AWS Proton environnement : fichier IaC Terraform

```
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
}
```

Infrastructure compilée sous forme de code

Lorsque vous créez un environnement ou un service, AWS Proton compile votre infrastructure sous forme de fichiers de code avec console ou spec file entrées. Il crée `proton.resource-type.variables.tf` des `proton.auto.tfvars.json` fichiers pour vos entrées qui peuvent être utilisés par Terraform, comme indiqué dans les exemples suivants. Ces fichiers se trouvent dans un référentiel spécifié dans un dossier qui correspond au nom de l'environnement ou de l'instance de service.

L'exemple montre comment AWS Proton inclure des balises dans la définition et les valeurs des variables, et comment vous pouvez propager ces AWS Proton balises aux ressources provisionnées. Pour de plus amples informations, veuillez consulter [the section called "Propagation des balises vers les ressources provisionnées"](#).

Exemple 2 : fichiers IaC compilés pour un environnement nommé « 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"
}
}

```

Chemins du référentiel

AWS Proton utilise les entrées de console ou de spécification provenant d'actions de création d'environnements ou de services pour trouver le référentiel et le chemin où se trouvent les fichiers iAC compilés. Les valeurs d'entrée sont transmises aux paramètres d'[entrée avec espace de noms](#).

AWS Proton prend en charge deux configurations de chemin de dépôt. Dans les exemples suivants, les chemins sont nommés selon les paramètres de ressources avec espace de noms provenant de deux environnements. Chaque environnement possède des instances de service de deux services, et les instances de service de l'un des services ont des composants directement définis.

Type de ressource	Paramètre de nom	=	Nom de la ressource
Environnement	environment.name		"env-prod"
Environnement	environment.name		"env-staged"
Service	service.name		"service-one"
Instance de service	service_instance.name	=	"instance-one-prod"
Instance de service	service_instance.name		"instance-one-staged"

Type de ressource	Paramètre de nom	=	Nom de la ressource
Service	<code>service.name</code>		<code>"service-two"</code>
Instance de service	<code>service_instance.name</code>		<code>"instance-two-prod"</code>
Composant	<code>service_instance.components.default.name</code>		<code>"component-prod"</code>
Instance de service	<code>service_instance.name</code>		<code>"instance-two-staged"</code>
Composant	<code>service_instance.components.default.name</code>		<code>"component-staged"</code>

Layout 1

S'il AWS Proton trouve le référentiel spécifié avec un `environments` dossier, il crée un dossier qui inclut les fichiers iAc compilés et porte le nom `environment.name`.

S'il AWS Proton trouve le référentiel spécifié avec un `environments` dossier contenant un nom de dossier correspondant à un nom d'environnement compatible avec une instance de service, il crée un dossier qui inclut les fichiers iAc de l'instance compilée et porte le nom `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

S'il AWS Proton trouve le référentiel spécifié sans `environments` dossier, il crée un `environment.name` dossier dans lequel il localise les fichiers iAc de l'environnement compilé.

S'il AWS Proton trouve le référentiel spécifié dont le nom de dossier correspond à un nom d'environnement compatible avec une instance de service, il crée un `service_instance.name` dossier dans lequel il localise les fichiers iAc de l'instance compilée.

```
/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
```

Fichier de schéma

En tant qu'administrateur, lorsque vous utilisez la [section Modèles de données d'API ouvertes \(schémas\)](#) pour définir un fichier YAML de schéma de paramètres pour votre ensemble de modèles, vous AWS Proton pouvez valider les entrées de valeurs de paramètres par rapport aux exigences que vous avez définies dans votre schéma.

Pour plus d'informations sur les formats et les mots clés disponibles, consultez la section relative aux [objets Schema](#) de l'OpenAPI.

Exigences relatives au schéma pour les ensembles de modèles d'environnement

Votre schéma doit suivre la [section Modèles de données \(schémas\)](#) de l'OpenAPI au format YAML. Il doit également faire partie de votre ensemble de modèles d'environnement.

Pour votre schéma d'environnement, vous devez inclure les en-têtes formatés pour indiquer que vous utilisez la section Modèles de données (schémas) de l'Open API. Dans les exemples de schéma d'environnement suivants, ces en-têtes apparaissent dans les trois premières lignes.

Un `environment_input_type` doit être inclus et défini avec un nom que vous fournissez. Dans les exemples suivants, cela est défini à la ligne 5. En définissant ce paramètre, vous l'associez à une ressource d' AWS Proton environnement.

Pour suivre le modèle de schéma Open API, vous devez inclure `types`. Dans l'exemple suivant, il s'agit de la ligne 6.

Ensuite `types`, vous devez définir un `environment_input_type` type. Vous définissez les paramètres d'entrée de votre environnement en tant que propriétés du `environment_input_type`. Vous devez inclure au moins une propriété dont le nom correspond à au moins un paramètre répertorié dans le fichier de code (IAC) de l'infrastructure environnementale associé au schéma.

Lorsque vous créez un environnement et que vous fournissez des valeurs de paramètres AWS Proton personnalisées, utilisez le fichier de schéma pour les associer, les valider et les injecter dans les paramètres frisés du fichier CloudFormation iAC associé. Pour chaque propriété (paramètre), indiquez un `name` et `type`. Facultativement, fournissez également un `description` `default`, et `pattern`.

Les paramètres définis pour l'exemple de schéma de modèle d'environnement standard suivant incluent `vpc_cidrsubnet_one_cidr`, et `subnet_two_cidr` avec le `default` mot-clé et les

valeurs par défaut. Lorsque vous créez un environnement avec ce schéma de bundle de modèles d'environnement, vous pouvez accepter les valeurs par défaut ou fournir les vôtres. Si un paramètre n'a pas de valeur par défaut et est répertorié en tant que `required` propriété (paramètre), vous devez lui fournir des valeurs lorsque vous créez un environnement.

Le deuxième exemple de schéma de modèle d'environnement standard répertorie le `required` paramètre `other_sample_input`.

Vous pouvez créer un schéma pour deux types de modèles d'environnement. Pour de plus amples informations, veuillez consulter [Enregistrer et publier des modèles](#).

- Modèles d'environnement standard

Dans l'exemple suivant, un type d'entrée d'environnement est défini avec une description et des propriétés d'entrée. Cet exemple de schéma peut être utilisé avec le fichier AWS Proton CloudFormation IaC présenté dans l'[exemple 3](#).

Exemple de schéma pour un modèle d'environnement standard :

```

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)))
```

Exemple de schéma pour un modèle d'environnement standard qui inclut un `required` paramètre :

```
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
```

- Modèles d'environnement gérés par le client

Dans l'exemple suivant, le schéma inclut uniquement une liste de sorties qui reproduisent les sorties de l'iAc que vous avez utilisé pour approvisionner votre infrastructure gérée par le client. Vous devez définir les types de valeurs de sortie sous forme de chaînes uniquement (et non sous forme de listes, de tableaux ou d'autres types). Par exemple, l'extrait de code suivant montre la section des sorties d'un modèle externe AWS CloudFormation . Cela provient du modèle présenté dans [l'exemple 1. Il peut être utilisé pour créer une infrastructure externe gérée par le client pour un service AWS Proton Fargate créé à partir de l'exemple 4.](#)

⚠ Important

En tant qu'administrateur, vous devez vous assurer que votre infrastructure provisionnée et gérée ainsi que tous les paramètres de sortie sont compatibles avec les modèles d'environnement gérés par le client associés. AWS Proton ne peut pas comptabiliser les modifications en votre nom, car elles ne sont pas visibles par AWS Proton. Les incohérences entraînent des échecs.

Exemples de sorties de fichiers CloudFormation IaC pour un modèle d'environnement géré par le client :

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

Le schéma du bundle de modèles d'environnement géré par le AWS Proton client correspondant est illustré dans l'exemple suivant. Chaque valeur de sortie est définie sous forme de chaîne.

Exemple de schéma pour un modèle d'environnement géré par le client :

```
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"
```

[...]

Exigences relatives au schéma pour les ensembles de modèles de services

Votre schéma doit suivre la [section Modèles de données \(schémas\)](#) de l'OpenAPI au format YAML, comme indiqué dans les exemples suivants. Vous devez fournir un fichier de schéma dans votre ensemble de modèles de services.

Dans les exemples de schéma de service suivants, vous devez inclure les en-têtes formatés. Dans l'exemple suivant, cela se trouve dans les trois premières lignes. Cela permet de vérifier que vous utilisez la section Modèles de données (schémas) de l'Open API.

A `service_input_type` doit être inclus et défini avec un nom que vous fournissez. Dans l'exemple suivant, il s'agit de la ligne 5. Cela associe les paramètres à une ressource AWS Proton de service.

Un pipeline AWS Proton de services est inclus par défaut lorsque vous utilisez la console ou la CLI pour créer un service. Lorsque vous incluez un pipeline de services pour votre service, vous devez l'inclure `pipeline_input_type` avec un nom que vous fournissez. Dans l'exemple suivant, il s'agit de la ligne 7. N'incluez pas ce paramètre si vous n'incluez pas de pipeline AWS Proton de services. Pour de plus amples informations, veuillez consulter [Enregistrer et publier des modèles](#).

Pour suivre le modèle de schéma Open API, vous devez inclure `types` dans l'exemple suivant, cela se trouve dans la ligne 9.

Ensuite `types`, vous devez définir un `service_input_type` type. Vous définissez les paramètres d'entrée de votre service en tant que propriétés de `service_input_type`. Vous devez inclure au moins une propriété dont le nom correspond à au moins un paramètre répertorié dans le fichier d'infrastructure de service sous forme de code (IAC) associé au schéma.

Pour définir un pipeline de services, en dessous de votre `service_input_type` définition, vous devez définir `unpipeline_input_type`. Comme ci-dessus, vous devez inclure au moins une propriété dont le nom correspond à au moins un paramètre répertorié dans un fichier iAc de pipeline associé au schéma. N'incluez pas cette définition si vous n'incluez pas de pipeline AWS Proton de services.

Lorsque, en tant qu'administrateur ou développeur, vous créez un service et que vous fournissez des valeurs de paramètres personnalisées, vous AWS Proton utilisez le fichier de schéma pour les faire correspondre, les valider et les injecter dans les paramètres frisés du fichier CloudFormation laC associé. Pour chaque propriété (paramètre), indiquez a name et type a. Facultativement, fournissez également un `descriptiondefault`, et `pattern`.

Les paramètres définis pour l'exemple de schéma incluent `portdesired_count`, `task_size` et `image` avec le `default` mot-clé et les valeurs par défaut. Lorsque vous créez un service avec ce schéma de bundle de modèles de services, vous pouvez accepter les valeurs par défaut ou fournir les vôtres. Le paramètre `unique_name` est également inclus dans l'exemple et ne possède pas de valeur par défaut. Il est répertorié en tant que `required` propriété (paramètre). En tant qu'administrateur ou développeur, vous devez fournir des valeurs pour `required` les paramètres lorsque vous créez des services.

Si vous souhaitez créer un modèle de service avec un pipeline de services, incluez-le `pipeline_input_type` dans votre schéma.

Exemple de fichier de schéma de service pour un service qui inclut un pipeline AWS Proton de services.

Cet exemple de schéma peut être utilisé avec les AWS Proton fichiers laC présentés dans les [exemples 4 et 5](#). Un pipeline de services est inclus.

```
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

```

Si vous souhaitez créer un modèle de service sans pipeline de services, ne l'incluez pas `pipeline_input_type` dans votre schéma, comme indiqué dans l'exemple suivant.

Exemple de fichier de schéma de service pour un service qui n'inclut pas de pipeline AWS Proton de services

```

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"

```

Résumez les fichiers modèles pour AWS Proton

Après avoir préparé les fichiers de votre environnement et de votre infrastructure de services sous forme de code (IaC) et leurs fichiers de schéma respectifs, vous devez les organiser dans des répertoires. Vous devez également créer un fichier manifeste YAML. Le fichier manifeste répertorie les fichiers iAc d'un répertoire, le moteur de rendu et le langage de modèle utilisé pour développer l'iAc dans ce modèle.

Note

Un fichier manifeste peut également être utilisé indépendamment des ensembles de modèles, en tant qu'entrée directe vers des composants directement définis. Dans ce cas, il spécifie toujours un seul fichier modèle IaC, à la fois pour Terraform CloudFormation et pour Terraform. Pour plus d'informations sur les composants, consultez [Composants](#).

Le fichier manifeste doit respecter le format et le contenu présentés dans l'exemple suivant.

CloudFormation format de fichier manifeste :

Avec CloudFormation, vous ne listez qu'un seul fichier.

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

Format de fichier manifeste Terraform :

Avec Terraform, vous pouvez répertorier explicitement un seul fichier ou utiliser le caractère générique * pour répertorier chacun des fichiers d'un répertoire.

Note

Le caractère générique inclut uniquement les fichiers dont le nom se termine par .tf. Les autres fichiers sont ignorés.

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

CodeBuildformat de fichier manifeste de provisionnement basé sur :

Avec le provisionnement CodeBuild basé, vous spécifiez les commandes shell de provisionnement et de déprovisionnement.

Note

Outre le manifeste, votre bundle doit inclure tous les fichiers dont dépendent vos commandes.

L'exemple de manifeste suivant utilise le provisionnement CodeBuild basé pour provisionner (déployer) et déprovisionner (détruire) les ressources à l'aide du AWS Cloud Development Kit (AWS CDK) (AWS CDK). Le bundle de modèles doit également inclure le code CDK.

Pendant le provisionnement, AWS Proton crée un fichier d'entrée contenant les valeurs des paramètres d'entrée que vous avez définis dans le schéma du modèle avec le nom `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 }}"
```

Après avoir configuré les répertoires et les fichiers manifestes pour votre environnement ou votre ensemble de modèles de services, vous compressez les répertoires dans une archive tar et vous les téléchargez dans un bucket Amazon Simple Storage Service (Amazon S3) AWS Proton où vous pouvez les récupérer, ou dans un référentiel Git de [synchronisation de modèles](#).

Lorsque vous créez une version mineure d'un modèle d'environnement ou de service auprès duquel vous vous êtes inscrit AWS Proton, vous indiquez le chemin d'accès à votre environnement ou à la base de modèles de services qui se trouve dans votre compartiment S3. AWS Proton l'enregistre avec la nouvelle version mineure du modèle. Vous pouvez sélectionner la nouvelle version mineure du modèle pour créer ou mettre à jour des environnements ou des services AWS Proton.

Résumé du bundle de modèles d'environnement

Il existe deux types d'ensembles de modèles d'environnement pour AWS Proton lesquels vous pouvez créer.

- Pour créer un ensemble de modèles d'environnement pour un modèle d'environnement standard, organisez le schéma, les fichiers d'infrastructure sous forme de code (IaC) et le fichier manifeste dans des répertoires, comme indiqué dans la structure de répertoires du bundle de modèles d'environnement suivante.
- Pour créer un ensemble de modèles d'environnement pour un modèle d'environnement géré par le client, fournissez uniquement le fichier et le répertoire du schéma. N'incluez pas le répertoire et les fichiers de l'infrastructure. AWS Proton renvoie une erreur si le répertoire et les fichiers de l'infrastructure sont inclus.

Pour de plus amples informations, veuillez consulter [Enregistrer et publier des modèles](#).

CloudFormation structure du répertoire du bundle de modèles d'environnement :

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

Structure du répertoire du bundle de modèles d'environnement Terraform :

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

Résumé de l'offre groupée de modèles de services

Pour créer un ensemble de modèles de services, vous devez organiser le schéma, les fichiers d'infrastructure en tant que code (IaC) et les fichiers manifestes dans des répertoires, comme indiqué dans l'exemple de structure de répertoire du bundle de modèles de services.

Si vous n'incluez pas de pipeline de services dans votre ensemble de modèles, n'incluez pas le répertoire et les fichiers du pipeline et ne définissez pas le modèle de service à associer à ce bundle de modèles "pipelineProvisioning": "CUSTOMER_MANAGED" lorsque vous créez le modèle de service.

Note

Vous ne pouvez pas le modifier une pipelineProvisioning fois le modèle de service créé.

Pour de plus amples informations, veuillez consulter [Enregistrer et publier des modèles](#).

CloudFormation structure du répertoire du bundle de modèles de services :

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

Structure du répertoire du bundle de modèles de services Terraform :

```
/schema
  schema.yaml
```

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

Considérations relatives aux ensembles de modèles

- Fichiers d'infrastructure sous forme de code (IaC)

AWS Proton modèles d'audits pour le format de fichier correct. Cependant, AWS Proton ne vérifie pas les erreurs de développement, de dépendance et de logique du modèle. Supposons, par exemple, que vous ayez spécifié la création d'un compartiment Amazon S3 dans votre fichier AWS CloudFormation IaC dans le cadre de votre modèle de service ou d'environnement. Un service est créé sur la base de ces modèles. Supposons maintenant que vous souhaitiez supprimer le service à un moment donné. Si le compartiment S3 spécifié n'est pas vide et que le fichier CloudFormation iAC ne le marque pas comme étant Retain dans leDeletionPolicy, AWS Proton échoue lors de l'opération de suppression du service.

- Limites de taille et format des fichiers groupés
 - Les limites de taille, de nombre et de taille de nom des fichiers groupés sont disponibles à l'adresse [Quotas AWS Proton](#).
 - Les répertoires de fichiers des ensembles de modèles sont compressés dans une archive tar et situés dans un compartiment Amazon Simple Storage Service (Amazon S3).
 - Chaque fichier du bundle doit être un fichier YAML formaté valide.
- Chiffrement groupé de modèles de compartiments S3

Si vous souhaitez chiffrer les données sensibles de vos ensembles de modèles au repos dans votre compartiment S3, utilisez les clés SSE-S3 ou SSE-KMS pour permettre de les récupérer.

AWS Proton

AWS Proton Modèles

Pour ajouter votre ensemble de modèles à votre bibliothèque de AWS Proton modèles, créez une version mineure du modèle et enregistrez-la auprès de AWS Proton. Lors de la création du modèle, indiquez le nom du compartiment Amazon S3 et le chemin de votre ensemble de modèles. Une fois les modèles publiés, ils peuvent être sélectionnés par les membres de l'équipe et les développeurs de la plateforme. Une fois sélectionnés, AWS Proton utilise le modèle pour créer et provisionner une infrastructure et des applications.

En tant qu'administrateur, vous pouvez créer et enregistrer un modèle d'environnement avec AWS Proton. Ce modèle d'environnement peut ensuite être utilisé pour déployer plusieurs environnements. Par exemple, il peut être utilisé pour déployer des environnements « de développement », « de test » et « de production ». L'environnement « dev » peut inclure un VPC avec des sous-réseaux privés et une politique d'accès restrictive à toutes les ressources. Les sorties d'environnement peuvent être utilisées comme entrées pour les services.

Vous pouvez créer et enregistrer des modèles d'environnement pour créer deux types d'environnements différents. Vous et les développeurs pouvez les utiliser avec AWS Proton pour déployer des services vers les deux types.

- Enregistrez et publiez un modèle d'environnement standard qui permet à AWS Proton de créer un environnement standard qui provisionne et gère l'infrastructure de l'environnement.
- Enregistrez et publiez un modèle d'environnement géré par le client avec AWS Proton qui permet de créer un environnement géré par le client qui se connecte à votre infrastructure provisionnée existante. AWS Proton ne gère pas votre infrastructure provisionnée existante.

Vous pouvez créer et enregistrer des modèles de service avec AWS Proton pour déployer des services dans des environnements. Un environnement avec AWS Proton doit être créé pour qu'un service puisse y être déployé.

La liste suivante décrit comment créer et gérer des modèles avec AWS Proton.

- (Facultatif) Préparez un rôle IAM pour contrôler l'accès des développeurs aux appels d'API AWS Proton et aux rôles de service IAM avec AWS Proton. Pour plus d'informations, veuillez consulter [la section appelée "Rôles IAM"](#).
- Composez un ensemble de modèles. Pour plus d'informations, veuillez consulter [Packs de modèles](#).

- Créez et enregistrez un modèle uneAWS Proton fois le bundle de modèles composé, compressé et enregistré dans un compartiment Amazon S3. Pour ce faire, vous pouvez utiliser ouAWS CLI.
- Testez et utilisez le modèle pour créer et gérer les ressourcesAWS Proton provisionnées après son enregistrementAWS Proton.
- Créez et gérez les versions principales et secondaires du modèle tout au long de sa durée de vie.

Vous pouvez gérer les versions des modèles manuellement ou à l'aide de configurations de synchronisation des modèles :

- Utilisez laAWS Proton console etAWS CLI créez une nouvelle version mineure ou majeure.
- [Créez une configuration de synchronisation des modèles](#) qui permet de créerAWS Proton automatiquement une nouvelle version mineure ou majeure lorsqu'elle détecte une modification de votre ensemble de modèles dans un référentiel que vous définissez.

Pour plus d'informations, consultez la [référence de l'API TheAWS Proton Service](#).

Rubriques


- [Modèles versionnés](#)
- [Enregistrer et publier des modèles](#)
- [Affichage de données du modèle](#)
- [Met à jour un modèle](#)
- [Suppression de modèles](#)
- [Configurations de synchronisation de modèles](#)
- [Configurations de synchronisation des services](#)

Modèles versionnés

En tant qu'administrateur ou membre d'une équipe de plateforme, vous définissez, créez et gérez une bibliothèque de modèles versionnés utilisés pour provisionner les ressources de l'infrastructure. Il existe deux types de versions de modèles : les versions mineures et les versions majeures.

- Versions mineures : modifications apportées au modèle dont le schéma est rétrocompatible. Ces modifications n'obligent pas le développeur à fournir de nouvelles informations lors de la mise à jour vers la nouvelle version du modèle.

Lorsque vous essayez d'apporter une modification de version mineure AWS Proton, faites de votre mieux pour déterminer si le schéma de la nouvelle version est rétrocompatible avec les versions mineures précédentes du modèle. Si le nouveau schéma n'est pas rétrocompatible, l'enregistrement de la nouvelle version mineure AWS Proton échoue.

 Note

La compatibilité est déterminée uniquement en fonction du schéma. AWS Proton ne vérifie pas si le fichier d'infrastructure du bundle de modèles en tant que code (IaC) est rétrocompatible avec les versions mineures précédentes. Par exemple, AWS Proton ne vérifie pas si le nouveau fichier IAC entraîne des modifications majeures pour les applications qui s'exécutent sur l'infrastructure provisionnée par une version mineure précédente du modèle.

- Versions principales : modifications apportées au modèle qui peuvent ne pas être rétrocompatibles. Ces modifications nécessitent généralement de nouvelles entrées de la part du développeur et impliquent souvent des modifications du schéma du modèle.

Vous pouvez parfois choisir de désigner une modification rétrocompatible comme version majeure en fonction du modèle opérationnel de votre équipe.

La manière de AWS Proton déterminer si une demande de version de modèle concerne une version mineure ou majeure dépend de la manière dont les modifications du modèle sont suivies :

- Lorsque vous demandez explicitement la création d'une nouvelle version de modèle, vous demandez une version majeure en spécifiant un numéro de version principale, et vous demandez une version secondaire en ne spécifiant pas de numéro de version principale.
- Lorsque vous utilisez la [synchronisation des modèles](#) (et que vous ne faites donc pas de demandes explicites de version de modèle), AWS Proton tente de créer de nouvelles versions mineures pour les modifications de modèle qui se produisent dans le fichier YAML existant. AWS Proton crée une version majeure lorsque vous créez un nouveau répertoire pour le nouveau changement de modèle (par exemple, passez de la version 1 à la version 2).

Note

L'enregistrement d'une nouvelle version mineure basé sur la synchronisation des modèles échoue toujours s'il est déterminé que la modification n'est pas rétrocompatible.

Lorsque vous publiez une nouvelle version d'un modèle, celle-ci devient la version recommandée s'il s'agit de la version majeure et mineure la plus élevée. Les nouvelles ressources AWS Proton sont créées à l'aide de la nouvelle version recommandée et AWS Proton invite les administrateurs à utiliser la nouvelle version et à mettre à jour les ressources existantes qui utilisent une version obsolète.

Enregistrer et publier des modèles

Vous pouvez enregistrer et publier des modèles d'environnement et de service avec AWS Proton, comme décrit dans les sections suivantes.

Vous pouvez créer une nouvelle version d'un modèle à l'aide de la console ou AWS CLI.

Vous pouvez également utiliser la console ou AWS CLI créer un modèle et [configurer une synchronisation de modèles](#) pour celui-ci. Cette configuration permet la synchronisation à partir des ensembles de modèles situés dans les référentiels Git enregistrés que vous avez définis. Chaque fois qu'un commit est envoyé à votre référentiel pour modifier l'un de vos ensembles de modèles, une nouvelle version mineure ou majeure de votre modèle est créée, si cette version n'existe pas déjà. Pour en savoir plus sur les prérequis et les exigences en matière de configuration de synchronisation des modèles, consultez [Configurations de synchronisation de modèles](#).

Enregistrer et publier des modèles d'environnement

Vous pouvez enregistrer et publier les types de modèles d'environnement suivants.

- Enregistrez et publiez un modèle d'environnement standard AWS Proton utilisé pour déployer et gérer l'infrastructure de l'environnement.
- Enregistrez et publiez un modèle d'environnement géré par le client AWS Proton qui vous permet de vous connecter à l'infrastructure provisionnée existante que vous gérez. AWS Proton ne gère pas votre infrastructure provisionnée existante.

⚠ Important

En tant qu'administrateur, assurez-vous que votre infrastructure provisionnée et gérée ainsi que tous les paramètres de sortie sont compatibles avec les modèles d'environnement gérés par le client associés. AWS Protonne peut pas prendre en compte les modifications en votre nom car ces modifications ne sont pas visibles pour AWS Proton. Les incohérences entraînent des incidents.

Vous pouvez AWS CLI utiliser la console ou.

AWS Management Console

Utilisez la console pour enregistrer et publier un nouveau modèle d'environnement.

1. Dans la [AWS Proton console](#), choisissez Modèles d'environnement.
2. Choisissez Créer un modèle d'environnement.
3. Sur la page Créer un modèle d'environnement, dans la section Options de modèle, choisissez l'une des deux options de modèle disponibles.
 - Créez un modèle pour le provisionnement de nouveaux environnements.
 - Créez un modèle pour utiliser l'infrastructure provisionnée que vous gérez.
4. Si vous avez choisi Créer un modèle pour la mise en service de nouveaux environnements, dans la section Source du bundle de modèles, choisissez l'une des trois options de source du bundle de modèles disponibles. Pour en savoir plus sur les exigences et les prérequis pour la synchronisation des modèles, consultez [Configurations de synchronisation de modèles](#).
 - Utilisez l'un de nos exemples de packs de modèles.
 - Utilisez votre propre ensemble de modèles.
 - [Synchronisez les modèles depuis Git](#).
5. Indiquez le chemin d'accès à un ensemble de modèles.
 - a. Si vous avez choisi Utiliser l'un de nos exemples de packs de modèles :

Dans la section Exemple de bundle de modèles, sélectionnez un exemple de bundle de modèles.
 - b. Si vous avez choisi Synchroniser les modèles depuis Git, dans la section Code source :

- i. Sélectionnez le référentiel pour la configuration de synchronisation de vos modèles.
 - ii. Entrez le nom de la branche du référentiel à partir de laquelle effectuer la synchronisation.
 - iii. (Facultatif) Entrez le nom d'un répertoire pour limiter la recherche de votre ensemble de modèles.
 - c. Sinon, dans la section Emplacement du bundle S3, indiquez le chemin d'accès à votre bundle de modèles.
6. Dans la section Détails du modèle.
- a. Saisissez un Template name (Nom de modèle).
 - b. (Facultatif) Entrez un Nom d'affichage.
 - c. (Facultatif) Entrez une description de modèle.
7. (Facultatif) Cochez la case Personnaliser les paramètres de chiffrement (avancés) dans la section Paramètres de cryptage pour fournir votre propre clé de cryptage.
8. (Facultatif) Dans la section Tags, choisissez Ajouter un nouveau tag et entrez une clé et une valeur pour créer un tag géré par le client.
9. Choisissez Créer un modèle d'environnement.

Vous êtes maintenant sur une nouvelle page qui affiche le statut et les détails de votre nouveau modèle d'environnement. Ces informations incluent une liste de balises AWS gérées par le client. AWS Proton génère automatiquement des balises AWS gérées pour vous lorsque vous créez des ressources AWS Proton. Pour plus d'informations, veuillez consulter [AWS Proton ressources et balises](#).

10. Le statut d'un nouveau modèle d'environnement commence à l'état de brouillon. Vous et les autres personnes autorisées pouvez le consulter et y accéder. Procédez à l'étape suivante pour mettre le modèle à disposition.
11. Dans la section Versions des modèles, cliquez sur le bouton radio situé à gauche de la version secondaire du modèle que vous venez de créer (1.0). Vous pouvez également choisir Publier dans l'alerte d'information et ignorer l'étape suivante.
12. Dans la section Versions des modèles, choisissez Publier.
13. Le statut du modèle passe à Publié. Comme il s'agit de la dernière version du modèle, il s'agit de la version recommandée.
14. Dans le volet de navigation, sélectionnez Modèles d'environnement pour afficher la liste de vos modèles d'environnement et leurs détails.

Utilisez la console pour enregistrer les nouvelles versions principales et mineures d'un modèle d'environnement.

Pour plus d'informations, veuillez consulter [Modèles versionnés](#).

1. Dans la [AWS Protonconsole](#), choisissez Environment Templates.
2. Dans la liste des modèles d'environnement, choisissez le nom du modèle d'environnement pour lequel vous souhaitez créer une version majeure ou mineure.
3. Dans la vue détaillée du modèle d'environnement, choisissez Créer une nouvelle version dans la section Versions des modèles.
4. Sur la page Créer une nouvelle version de modèle d'environnement, dans la section Source du bundle de modèles, choisissez l'une des deux options de source du bundle de modèles disponibles.
 - Utilisez l'un de nos exemples de packs de modèles.
 - Utilisez votre propre ensemble de modèles.
5. Fournissez un chemin d'accès au bundle de modèles sélectionné.
 - Si vous avez choisi Utiliser l'un de nos exemples de lots de modèles, dans la section Ensemble d'exemples de modèles, sélectionnez un exemple de bundle de modèles.
 - Si vous avez choisi Utiliser votre propre ensemble de modèles, dans la section Emplacement du bundle S3, choisissez le chemin d'accès à votre ensemble de modèles.
6. Dans la section Détails du modèle.
 - a. (Facultatif) Entrez un Nom d'affichage.
 - b. (Facultatif) Entrez une description de modèle.
7. Dans la section Détails du modèle, choisissez l'une des options suivantes.
 - Pour créer une version mineure, laissez la case à cocher Cocher pour créer une nouvelle version majeure vide.
 - Pour créer une version majeure, cochez la case Cocher pour créer une nouvelle version majeure.
8. Suivez les étapes de la console pour créer la nouvelle version mineure ou majeure et choisissez Créer une nouvelle version.

AWS CLI

Utilisez l'interface de ligne de commande pour enregistrer et publier un nouveau modèle d'environnement, comme indiqué dans les étapes suivantes.

1. Créez un modèle d'environnement standard OU géré par le client en spécifiant la région, le nom, le nom d'affichage (facultatif) et la description (facultatif).
 - a. Créez un modèle d'environnement standard.

Exécutez la commande suivante :

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

Réponse :

```
{  
  "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. Créez un modèle d'environnement géré par le client en ajoutant le provisioning paramètre avec une valeur CUSTOMER_MANAGED.

Exécutez la commande suivante :

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

Réponse :

```
{
  "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. Créez une version mineure 0 de la version majeure 1 du modèle d'environnement

Cette étape et les autres sont identiques pour les modèles d'environnement standard et gérés par le client.

Incluez le nom du modèle, la version principale, le nom du compartiment S3 et la clé du compartiment qui contient votre ensemble de modèles d'environnement.

Exécutez la commande suivante :

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

Réponse :

```
{
  "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. Utilisez la commande `get` pour vérifier l'état des inscriptions.

Exécutez la commande suivante :

```

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

```

Réponse :

```

{
  "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"
  }
}

```

- Publier la version mineure 0 de la version majeure 1 du modèle d'environnement en fournissant le nom du modèle et les versions principale et mineure. Cette version est la Recommended version.

Exécutez la commande suivante :

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

Réponse :

```
{
  "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"
  }
}
```

Après avoir créé un nouveau modèle à l'aide du AWS CLI, vous pouvez consulter la liste AWS des balises gérées par le client. AWS Proton génère automatiquement des balises AWS gérées pour

vous. Vous pouvez également modifier et créer des balises gérées par le client à l'aide du AWS CLI. Pour plus d'informations, veuillez consulter [AWS Proton ressources et balises](#).

Exécutez la commande suivante :

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

Enregistrer et publier des modèles de services

Lorsque vous créez une version de modèle de service, vous spécifiez une liste de modèles d'environnement. Ainsi, lorsque les développeurs sélectionnent un modèle de service, ils peuvent choisir l'environnement dans lequel déployer leur service.

Avant de créer un service à partir d'un modèle de service ou avant de publier un modèle de service, vérifiez que les environnements sont déployés à partir des modèles d'environnement compatibles répertoriés.

Vous ne pouvez pas mettre à jour un service vers la nouvelle version majeure s'il est déployé dans un environnement créé à partir d'un modèle d'environnement compatible supprimé.

Pour ajouter ou supprimer des modèles d'environnement compatibles pour une version de modèle de service, vous devez en créer une nouvelle version majeure.

Vous pouvez AWS CLI utiliser la console ou.

AWS Management Console

Utilisez la console pour enregistrer et publier un nouveau modèle de service.

1. Dans la [AWS Proton console](#), choisissez Modèles de services.
2. Choisissez Créer un modèle de service.
3. Sur la page Créer un modèle de service, dans la section Source du bundle de modèles, choisissez l'une des options de modèle disponibles.
 - Utilisez votre propre ensemble de modèles.
 - Synchronisez les modèles depuis Git.

4. Indiquez le chemin d'accès à un ensemble de modèles.
 - a. Si vous avez choisi Synchroniser les modèles depuis Git, dans la section Dépôt de code source :
 - i. Sélectionnez le référentiel pour la configuration de synchronisation de vos modèles.
 - ii. Entrez le nom de la branche du référentiel à partir de laquelle effectuer la synchronisation.
 - iii. (Facultatif) Entrez le nom d'un répertoire pour limiter la recherche de votre ensemble de modèles.
 - b. Sinon, dans la section Emplacement du bundle S3, indiquez le chemin d'accès à votre bundle de modèles.
5. Dans la section Détails du modèle.
 - a. Saisissez un Template name (Nom de modèle).
 - b. (Facultatif) Entrez un Nom d'affichage.
 - c. (Facultatif) Entrez une description de modèle.
6. Dans la section Modèles d'environnement compatibles, choisissez parmi une liste de modèles d'environnement compatibles.
7. (Facultatif) Dans la section Paramètres de chiffrement, choisissez Personnaliser les paramètres de chiffrement (avancés) pour fournir votre propre clé de cryptage.
8. (Facultatif) Dans la section Pipeline :

Si vous n'incluez aucune définition de pipeline de services dans votre modèle de service, décochez la case Pipeline - facultatif en bas de la page. Vous ne pouvez pas modifier cela une fois le modèle de service créé. Pour plus d'informations, veuillez consulter [Packs de modèles](#).

9. (Facultatif) Dans la section Sources de composants prises en charge, pour Sources de composants, sélectionnez Directement défini pour permettre l'attachement de composants directement définis à vos instances de service.
10. (Facultatif) Dans la section Tags, choisissez Ajouter un nouveau tag et entrez une clé et une valeur pour créer un tag géré par le client.
11. Choisissez Créer un modèle de service.

Vous êtes maintenant sur une nouvelle page qui affiche le statut et les détails de votre nouveau modèle de service. Ces informations incluent une liste deAWS balises gérées par le client. AWS Protongénère automatiquement des balisesAWS gérées pour vous lorsque vous créez desAWS Proton ressources. Pour plus d'informations, veuillez consulter [AWS Protonressources et balises](#).

12. Le statut d'un nouveau modèle de service commence à l'état Brouillon. Vous et les autres personnesproton:CreateService autorisées pouvez le consulter et y accéder. Procédez à l'étape suivante pour mettre le modèle à disposition.
13. Dans la section Versions des modèles, cliquez sur le bouton radio situé à gauche de la version secondaire du modèle que vous venez de créer (1.0). Vous pouvez également choisir Publier dans l'alerte d'information et ignorer l'étape suivante.
14. Dans la section Versions des modèles, choisissez Publier.
15. Le statut du modèle passe à Publié. Comme il s'agit de la dernière version du modèle, il s'agit de la version recommandée.
16. Dans le volet de navigation, sélectionnez Modèles de services pour afficher la liste de vos modèles de service et leurs détails.

Utilisez la console pour enregistrer les nouvelles versions principales et mineures d'un modèle de service.

Pour plus d'informations, veuillez consulter [Modèles versionnés](#).

1. Dans la [AWS Protonconsole](#), choisissez Modèles de services.
2. Dans la liste des modèles de service, choisissez le nom du modèle de service pour lequel vous souhaitez créer une version principale ou secondaire.
3. Dans la vue détaillée du modèle de service, choisissez Créer une nouvelle version dans la section Versions des modèles.
4. Sur la page Créer une nouvelle version de modèle de service, dans la section Source du bundle, sélectionnez Utiliser votre propre bundle de modèles.
5. Dans la section Emplacement du bundle S3, choisissez le chemin d'accès à votre bundle de modèles.
6. Dans la section Détails du modèle.
 - a. (Facultatif) Entrez un Nom d'affichage.

- b. (Facultatif) Entrez une description de modèle.
7. Dans la section Détails du modèle, choisissez l'une des options suivantes.
 - Pour créer une version mineure, laissez la case à cocher Cocher pour créer une nouvelle version majeure vide.
 - Pour créer une version majeure, cochez la case Cocher pour créer une nouvelle version majeure.
8. Suivez les étapes de la console pour créer la nouvelle version mineure ou majeure et choisissez Créer une nouvelle version.

AWS CLI

Pour créer un modèle de service qui déploie un service sans pipeline de services, ajoutez le paramètre et la valeur `--pipeline-provisioning "CUSTOMER_MANAGED"` à la `create-service-template` commande. Configurez vos ensembles de modèles comme décrit dans [Packs de modèles](#) Création et [Exigences relatives au schéma pour les ensembles de modèles de services](#).

Note

Vous ne pouvez pas le modifier une `pipelineProvisioning` fois le modèle de service créé.

1. Utilisez l'interface de ligne de commande pour enregistrer et publier un nouveau modèle de service, avec ou sans pipeline de services, comme indiqué dans les étapes suivantes.
 - a. Créez un modèle de service avec un pipeline de services à l'aide de l'interface de ligne de commande.

Spécifiez le nom, le nom d'affichage (facultatif) et la description (facultatif).

Exécutez la commande suivante :

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

Réponse :

```
{
  "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. Créez un modèle de service sans pipeline de services.

Addition `--pipeline-provisioning`.

Exécutez la commande suivante :

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

Réponse :

```
{
  "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. Créez une version mineure 0 de la version majeure 1 du modèle de service.

Incluez le nom du modèle, les modèles d'environnement compatibles, la version principale, ainsi que le nom du compartiment S3 et la clé du compartiment qui contient votre ensemble de modèles de service.

Exécutez la commande suivante :

```
$ 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"} ]'
```

Réponse :

```
{  
  "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. Utilisez la commande get pour vérifier l'état des inscriptions.

Exécutez la commande suivante :

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

Réponse :

```
{
  "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\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. Publiez le modèle de service à l'aide de la commande de mise à jour pour modifier le statut en "PUBLISHED".

Exécutez la commande suivante :

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

Réponse :

```
{
  "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. Vérifiez que la version 1.0 AWS Proton a été publiée en utilisant la commande `get` pour récupérer les données détaillées du modèle de service.

Exécutez la commande suivante :

```

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

```

Réponse :

```

{
  "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\"
}
}

```

Affichage de données du modèle

Vous pouvez consulter des listes de modèles avec des détails et afficher des modèles individuels avec des données détaillées à l'aide de la [AWS Protonconsole](#) et AWS CLI.

Les données du modèle d'environnement géré par le client incluent le `provisioned` paramètre avec la valeur `CUSTOMER_MANAGED`.

Si un modèle de service n'inclut pas de pipeline de services, les données du modèle de service incluent le `pipelineProvisioning` paramètre avec la valeur `CUSTOMER_MANAGED`.

Pour plus d'informations, veuillez consulter [Enregistrer et publier des modèles](#).

Vous pouvez utiliser la console ou le AWS CLI pour répertorier et afficher les données du modèle.

AWS Management Console

Utilisez la console pour répertorier et afficher les modèles.

1. Pour afficher la liste des modèles, choisissez des modèles (Environnement ou Service).
2. Pour afficher les données détaillées, choisissez le nom d'un modèle.

Consultez les données détaillées du modèle, la liste des versions principales et secondaires du modèle, la liste des AWS Proton ressources qui ont été déployées à l'aide de versions de modèle et de balises de modèle.

La version majeure et la version mineure recommandées sont étiquetées comme recommandées.

AWS CLI

Utilisez leAWS CLI pour répertorier et afficher les modèles.

Exécutez la commande suivante :

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

Réponse :

```
{  
  "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"  
  }  
}
```

Exécutez la commande suivante :

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

Réponse :

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

Afficher une version secondaire d'un modèle de service.

Exécutez la commande suivante :

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

Réponse :

```
{
  "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"
  }
}

```

Affichez un modèle de service sans pipeline de services, comme indiqué dans l'exemple de commande et de réponse suivant.

Exécutez la commande suivante :

```

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

```

Réponse :

```
{
  "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"
  }
}
```

Met à jour un modèle

Vous pouvez mettre à jour un modèle comme décrit dans la liste suivante.

- Modifiez le `description` ou `displayName` d'un modèle lorsque vous utilisez la console ou AWS CLI. Vous ne pouvez pas modifier le `name` d'un modèle.
- Mettez à jour le statut de la version mineure d'un modèle lorsque vous utilisez la console ou AWS CLI. Vous pouvez uniquement modifier le statut de `DRAFT` à `PUBLISHED`.
- Modifiez le nom d'affichage et la description d'une version mineure ou majeure d'un modèle lorsque vous utilisez le AWS CLI.

AWS Management Console

Modifiez la description et le nom d'affichage d'un modèle à l'aide de la console, comme décrit dans les étapes suivantes.

Dans la liste des modèles.

1. Dans la [AWS Proton console](#), choisissez Modèles (Environnement ou Service).
2. Dans la liste des modèles, cliquez sur le bouton radio situé à gauche du modèle dont vous souhaitez mettre à jour la description ou le nom d'affichage.
3. Choisissez Actions, puis Modifier.

4. Sur la page Modifier le modèle (environnement ou service), dans la section Détails du modèle, saisissez vos modifications dans le formulaire et choisissez Enregistrer les modifications.

Modifiez le statut d'une version mineure d'un modèle à l'aide de la console pour publier un modèle, comme décrit ci-dessous. Vous pouvez uniquement modifier le statut de DRAFT à PUBLISHED.

Dans la page détaillée du modèle (environnement ou service).

1. Dans la [AWS Proton console](#), choisissez des modèles (Environnement ou Service).
2. Dans la liste des modèles, choisissez le nom du modèle dont vous souhaitez mettre à jour le statut d'une version mineure de Brouillon à Publié.
3. Sur la page détaillée du modèle (environnement ou service), dans la section Versions des modèles, sélectionnez le bouton radio situé à gauche de la version secondaire que vous souhaitez publier.
4. Choisissez Publier dans la section Versions des modèles. Le statut passe de Brouillon à Publié.

AWS CLI

L'exemple de commande et de réponse suivant montre comment modifier la description d'un modèle d'environnement.

Exécutez la commande suivante .

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

Réponse :

```
{  
  "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"
  }
}

```

Vous pouvez également utiliser le AWS CLI pour mettre à jour les modèles de service. [Enregistrer et publier des modèles de services](#) Reportez-vous à l'étape 5 pour un exemple de mise à jour du statut d'une version mineure d'un modèle de service.

Suppression de modèles

Les modèles peuvent être supprimés à l'aide de la console et AWS CLI.


Vous pouvez supprimer une version mineure d'un modèle d'environnement si aucun environnement n'est déployé sur cette version.

Vous pouvez supprimer une version mineure d'un modèle de service si aucune instance de service ou aucun pipeline n'est déployé sur cette version. Votre pipeline peut être déployé sur une version de modèle différente de celle de votre instance de service. Par exemple, si votre instance de service est mise à jour vers la version 1.1 à partir de la version 1.0 et que votre pipeline est toujours déployé vers la version 1.0, vous ne pouvez pas supprimer le modèle de service 1.0.

AWS Management Console

Vous pouvez utiliser la console pour supprimer l'intégralité du modèle ou des versions mineures et majeures individuelles d'un modèle.

Utilisez la console pour supprimer des modèles.

 Note

Lorsque vous utilisez la console pour supprimer des modèles.

- Lorsque vous supprimez l'intégralité du modèle, vous supprimez également les versions principales et secondaires du modèle.

Dans la liste des modèles (d'environnement ou de service).

1. Dans la [AWS Protonconsole](#), choisissez Modèles (Environnement ou Service).
2. Dans la liste des modèles, activez la case d'option.

Vous ne pouvez supprimer un modèle complet que si aucuneAWS Proton ressource n'est déployée sur ses versions.

3. Choisissez Actions, puis Supprimer pour supprimer l'intégralité du modèle.
4. Un mode vous demande de confirmer l'action de suppression.
5. Suivez les instructions et choisissez Oui, supprimer.

Dans la page détaillée du modèle (environnement ou service).

1. Dans la [AWS Protonconsole](#), choisissez Modèles (Environnement ou Service).
2. Dans la liste des modèles, choisissez le nom du modèle à supprimer entièrement ou supprimez des versions principales ou secondaires.
3. Pour supprimer l'intégralité du modèle.

Vous ne pouvez supprimer un modèle complet que si aucuneAWS Proton ressource n'est déployée sur ses versions.

- a. Choisissez Supprimer dans le coin supérieur droit de la page.
 - b. Un mode vous demande de confirmer l'action de suppression.
 - c. Suivez les instructions et choisissez Oui, supprimer.
4. Supprimer les versions principales ou secondaires d'un modèle.

Vous ne pouvez supprimer une version mineure d'un modèle que si aucuneAWS Proton ressource n'est déployée sur cette version.

- a. Dans la section Versions de modèle, activez la case d'option.
- b. Choisissez Supprimer dans la section Versions du modèle.
- c. Un mode vous demande de confirmer l'action de suppression.
- d. Suivez les instructions et choisissez Oui, supprimer.

AWS CLI

AWS CLI Les opérations de suppression de modèles n'incluent pas la suppression d'autres versions d'un modèle. Lorsque vous utilisez le AWS CLI, supprimez les modèles en respectant les conditions suivantes.

- Supprimez un modèle entier s'il n'existe aucune version mineure ou majeure du modèle.
- Supprimez une version majeure lorsque vous supprimez la dernière version mineure restante.
- Supprimez une version secondaire d'un modèle si aucune AWS Proton ressource n'est déployée sur cette version.
- Supprimez la version mineure recommandée d'un modèle s'il n'existe aucune autre version mineure du modèle et si aucune AWS Proton ressource n'est déployée vers cette version.

Les exemples de commandes et de réponses suivants montrent comment utiliser les modèles AWS CLI pour supprimer des modèles.

Exécutez la commande suivante :

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

Réponse :

```
{  
  "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"
  }
}
```

Exécutez la commande suivante :

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

Réponse :

```
{
  "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"
  }
}
```

Exécutez la commande suivante :

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

Réponse :

```
{
  "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"
  }
}
```

Configurations de synchronisation de modèles

Découvrez comment configurer un modèle pour permettre AWS Proton synchronisation à partir de ensembles de modèles situés dans les référentiels git enregistrés que vous définissez. Lorsqu'un commit est envoyé vers votre dépôt, AWS Proton vérifie les modifications apportées aux ensembles de modèles de votre référentiel. S'il détecte un changement de bundle de modèles, une nouvelle version mineure ou majeure de son modèle est créée, si cette version n'existe pas déjà. AWS Proton prend actuellement en charge GitHub, GitHub Enterprise, et BitBucket.

Transférer un commit vers un ensemble de modèles synchronisés

Lorsque vous envoyez un commit à une branche qui est suivie par l'un de vos modèles, AWS Proton clone votre dépôt et détermine les modèles à synchroniser. Il analyse les fichiers de votre répertoire pour trouver les répertoires correspondant à la convention de `{template-name}/` `{major-version}/`.

Après AWS Proton détermine quels modèles et versions principales sont associés à votre référentiel et à votre branche, il commence à essayer de synchroniser tous ces modèles en parallèle.

Lors de chaque synchronisation avec un modèle particulier, AWS Proton vérifie d'abord si le contenu du répertoire des modèles a changé depuis la dernière synchronisation réussie. Si le contenu n'a pas changé, AWS Proton ignore l'enregistrement d'un bundle dupliqué. Cela garantit qu'une nouvelle version mineure du modèle est créée si le contenu du bundle de modèles change. Si le contenu du bundle de modèles a changé, le bundle est enregistré auprès de AWS Proton.

Une fois le bundle de modèles enregistré, AWS Proton surveille le statut de l'enregistrement jusqu'à ce que celui-ci soit terminé.

Une seule synchronisation peut avoir lieu entre une version mineure et une version majeure du modèle à la fois. Toutes les validations qui auraient pu être poussées alors qu'une synchronisation était en cours sont regroupées par lots. Les validations par lots sont synchronisées une fois la tentative de synchronisation précédente terminée.

Modèles de service de synchronisation

AWS Proton peut synchroniser les modèles d'environnement et de service depuis votre dépôt git. Pour synchroniser vos modèles de service, vous ajoutez un fichier supplémentaire nommé `template-registration.yaml` dans chaque répertoire des versions majeures de votre ensemble de modèles. Ce fichier contient des informations supplémentaires qui AWS Proton besoins lorsqu'il crée une version de modèle de service pour vous à la suite d'un commit : environnements compatibles et sources de composants prises en charge.

Le chemin complet du fichier est `service-template-name/major-version/.template-registration.yaml`. Pour plus d'informations, veuillez consulter [the section called "Modèles de service de synchronisation"](#).

Considérations relatives à la configuration de la synchronisation

Passez en revue les considérations suivantes concernant l'utilisation des configurations de synchronisation de modèles.

- Les référentiels ne doivent pas dépasser 250 Mo.
- Pour configurer la synchronisation des modèles, associez d'abord le référentiel à AWS Proton. Pour plus d'informations, veuillez consulter [the section called "Création d'un lien vers le référentiel"](#).
- Lorsqu'une nouvelle version de modèle est créée à partir d'un modèle synchronisé, elle se trouve dans DRAFT état.
- Une nouvelle version mineure d'un modèle est créée si l'une des conditions suivantes est vraie :
 - Le contenu du bundle de modèles est différent de celui de la dernière version mineure du modèle synchronisé.
 - La dernière version mineure du modèle précédemment synchronisée a été supprimée.
- La synchronisation ne peut pas être interrompue.
- Les nouvelles versions mineures ou majeures sont automatiquement synchronisées.
- Les nouveaux modèles de haut niveau ne peuvent pas être créés par des configurations de synchronisation de modèles.

- Vous ne pouvez pas effectuer de synchronisation avec un modèle à partir de plusieurs référentiels avec une configuration de synchronisation de modèles.
- Vous ne pouvez pas utiliser de balises à la place de branches.
- Lorsque vous [créer un modèle de service](#), vous spécifiez des modèles d'environnement compatibles.
- Vous pouvez créer un modèle d'environnement et l'ajouter en tant qu'environnement compatible pour votre modèle de service dans le même commit.
- Les synchronisations avec une seule version majeure du modèle sont exécutées une par une. Au cours d'une synchronisation, si de nouveaux commits sont détectés, ils sont regroupés par lots et appliqués à la fin de la synchronisation active. Les synchronisations avec les différentes versions majeures du modèle se font en parallèle.
- Si vous modifiez la branche à partir de laquelle vos modèles sont synchronisés, toutes les synchronisations en cours depuis l'ancienne branche sont d'abord terminées. La synchronisation commence ensuite à partir de la nouvelle branche.
- Si vous modifiez le référentiel à partir duquel vos modèles sont synchronisés, toute synchronisation en cours depuis l'ancien référentiel risque d'échouer ou de s'exécuter jusqu'à la fin. Cela dépend de l'étape de synchronisation dans laquelle ils se trouvent.

Pour plus d'informations, consultez le [LeAWS ProtonRéférence de l'API de service](#).

Rubriques

- [Création d'une configuration de synchronisation de modèles](#)
- [Afficher les détails de configuration de synchronisation des modèles](#)
- [Modifier la configuration de synchronisation d'un modèle](#)
- [Supprimer un modèle de configuration de synchronisation](#)

Création d'une configuration de synchronisation de modèles

Découvrez comment créer un modèle de configuration de synchronisation avecAWS Proton.

Créez un modèle de configuration prérequis pour la synchronisation :

- Vous avez [lié à un dépôt](#) avecAWS Proton.
- UN [ensemble de modèles](#) se trouve dans votre dépôt.

Le lien vers le référentiel est le suivant :

- Un `CodeConnectionsconnexion` qui donne AWS Proton autorisation d'accéder à votre dépôt et de vous abonner à ses notifications.
- Un [rôle lié au service](#). Lorsque vous liez votre référentiel, le rôle lié au service est créé pour vous.

Avant de créer votre première configuration de synchronisation de modèles, transférez un ensemble de modèles vers votre référentiel, comme indiqué dans le schéma de répertoire suivant.

```

/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/

```

Une fois que vous avez créé votre première configuration de synchronisation de modèles, de nouvelles versions de modèles sont automatiquement créées lorsque vous envoyez un commit qui ajoute un ensemble de modèles mis à jour dans une nouvelle version (par exemple, sous `/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/

```

Vous pouvez inclure de nouvelles versions de bundle de modèles pour un ou plusieurs modèles configurés pour la synchronisation dans un seul commit. AWS Proton crée une nouvelle version de modèle pour chaque nouvelle version de bundle de modèles incluse dans le commit.

Après avoir créé la configuration de synchronisation du modèle, vous pouvez toujours créer manuellement de nouvelles versions du modèle dans la console ou à l'aide du `AWS CLI` en téléchargeant des ensembles de modèles depuis un compartiment S3. La synchronisation des modèles ne fonctionne que dans un seul sens : de votre dépôt vers AWS Proton. Versions de modèles créées manuellement ne sont pas synchronisées.

Après avoir configuré une configuration de synchronisation des modèles, AWS Proton écoute les modifications apportées à votre dépôt. Chaque fois qu'une modification est poussée, elle recherche tout répertoire portant le même nom que votre modèle. Il recherche ensuite dans ce répertoire tous les répertoires qui ressemblent à des versions majeures. AWS Proton enregistre le bundle de modèles dans la version principale du modèle correspondant. Les nouvelles versions se trouvent toujours dans le DRAFT état. Tu peux [publier les nouvelles versions](#) avec la console ou AWS CLI.

Supposons, par exemple, que vous disposiez d'un modèle appelé `my-env-template` configuré pour être synchronisé à partir de `my-repo/template` sur une succursale `main` avec la mise en page suivante.

```
/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 synchronise le contenu de `/templates/my-env-template/v1/` pour `my-env-template:1` et le contenu de `/templates/my-env-template/v2/` pour `my-env-template:2`. S'ils n'existent pas déjà, il crée ces versions majeures.

AWS Proton a trouvé le premier répertoire correspondant au nom du modèle. Vous pouvez limiter les répertoires que AWS Proton recherche en spécifiant `subdirectoryPath` lorsque vous créez ou modifiez une configuration de synchronisation de modèles. Par exemple, vous pouvez spécifier `production-templates/` pour `subdirectoryPath`.

Vous pouvez créer un modèle de configuration de synchronisation à l'aide de la console ou de l'interface de ligne de commande.

AWS Management Console

Créez un modèle et une configuration de synchronisation de modèles à l'aide de la console.

1. Dans le [AWS Proton console](#), choisissez `Modèles d'environnement`.

2. Choisissez **Création d'un modèle d'environnement**.
3. Dans le **Création d'un modèle d'environnement** page, dans le **Options du modèle** section, choisissez **Création d'un modèle pour le provisionnement de nouveaux environnements**.
4. Dans le **Source du bundle de modèles** section, choisissez **Synchroniser les modèles depuis Git**.
5. Dans le **Référentiel de code source** section :
 - a. Pour **Référentiel**, sélectionnez le référentiel lié qui contient votre ensemble de modèles.
 - b. Pour **Succursale**, sélectionnez une branche du référentiel à partir de laquelle effectuer la synchronisation.
 - c. (Facultatif) Pour **Répertoire des ensembles de modèles**, entrez le nom d'un répertoire pour affiner la recherche de votre ensemble de modèles.
6. Dans le **Détails du modèle** section.
 - a. Saisissez un **Template name** (Nom de modèle).
 - b. (Facultatif) Entrez un **Nom d'affichage du modèle**.
 - c. (Facultatif) Entrez un **Description du modèle** pour le modèle d'environnement.
7. (Facultatif) Cochez la case pour **Personnaliser les paramètres de chiffrement (avancés)** dans le **Paramètres de chiffrement** section pour fournir votre propre clé de chiffrement.
8. (Facultatif) Dans le **Balises** section, choisissez **Ajouter un nouveau tag** entrez une clé et une valeur pour créer un tag géré par le client.
9. Choisissez **Créer un modèle d'environnement**.

Vous êtes maintenant sur une nouvelle page qui affiche le statut et les détails de votre nouveau modèle d'environnement. Ces informations incluent une liste de **AWS tags** gérés et gérés par le client. **AWS Proton** génère automatiquement **AWS** des tags gérés pour vous lorsque vous créez **AWS Proton** ressources. Pour plus d'informations, veuillez consulter [AWS Proton ressources et balises](#).

10. Dans la page détaillée du modèle, choisissez le **Synchronisation** onglet pour afficher les données détaillées de configuration de synchronisation des modèles.
11. Choisissez le **Versions du modèle** onglet pour afficher les versions du modèle avec les détails du statut.
12. L'état d'un nouveau modèle d'environnement commence dans le **Brouillon** état. Vous et d'autres personnes avec `proton:CreateEnvironment` les autorisations peuvent le consulter et y accéder. Suivez l'étape suivante pour mettre le modèle à la disposition des autres utilisateurs.

13. Dans le Versions du modèle section, cliquez sur le bouton radio situé à gauche de la version mineure du modèle que vous venez de créer (1.0). Comme alternative, vous pouvez choisir Publier dans l'alerte d'information et sautez l'étape suivante.
14. Dans le Versions du modèle section, choisissez Publier.
15. Le statut du modèle passe à Publié. C'est le dernier en date et Recommandé version du modèle.
16. Dans le volet de navigation, sélectionnez Modèles d'environnement pour afficher la liste des modèles et des détails de votre environnement.

La procédure de création d'un modèle de service et d'une configuration de synchronisation de modèles est similaire.

AWS CLI

Créez un modèle et une configuration de synchronisation de modèles à l'aide du AWS CLI.

1. Créez un modèle. Dans cet exemple, un modèle d'environnement est créé.

Exécutez la commande suivante .

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

La réponse est la suivante.

```
{  
  "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. Créez la configuration de synchronisation de votre modèle avec AWS CLI en fournissant les éléments suivants :
 - Le modèle avec lequel vous souhaitez effectuer la synchronisation. Après avoir créé la configuration de synchronisation du modèle, vous pouvez toujours créer de nouvelles versions à partir de celui-ci manuellement dans la console ou à l'aide de l'AWS CLI.
 - Nom du modèle.
 - Type de modèle.
 - Le référentiel lié à partir duquel vous souhaitez effectuer la synchronisation.
 - Le fournisseur du référentiel lié.
 - Branche dans laquelle se trouve le bundle de modèles.
 - (Facultatif) Le chemin d'accès au répertoire contenant votre ensemble de modèles. Par défaut, AWS Proton recherche le premier répertoire correspondant au nom de votre modèle.

Exécutez la commande suivante .

```
$ 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/"
```

La réponse est la suivante.

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

3. Pour publier la version de votre modèle, voir [Enregistrer et publier des modèles](#).

Modèles de service de synchronisation

Les exemples précédents montrent comment synchroniser les modèles d'environnement. Les modèles de service sont similaires. Pour synchroniser les modèles de service, vous ajoutez un fichier supplémentaire nommé `template-registration.yaml` dans chaque répertoire des versions majeures de votre ensemble de modèles. Ce fichier contient des informations supplémentaires qui AWS Proton est nécessaire lorsqu'il crée une version de modèle de service pour vous après un commit. Lorsque vous créez explicitement une version de modèle de service à l'aide du AWS Proton console ou API, vous fournissez ces informations sous forme d'entrées, et ce fichier remplace ces entrées pour la synchronisation des modèles.

```
./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/
```

Le `template-registration.yaml` le fichier contient les informations suivantes :

- Environnements compatibles[obligatoire] — Les environnements basés sur ces modèles d'environnement et les versions majeures sont compatibles avec les services basés sur cette version de modèle de service.
- Sources de composants prises en charge[facultatif] — Les composants utilisant ces sources sont compatibles avec les services basés sur cette version de modèle de service. Si ce n'est pas spécifié, les composants ne peuvent pas être attachés à ces services. Pour plus d'informations sur les composants, voir [Composants](#).

La syntaxe YAML du fichier est la suivante :

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

Spécifiez une ou plusieurs combinaisons de modèles d'environnement et de versions principales. Spécifications `supported_component_sources` est facultatif, et la seule valeur prise en charge est `DIRECTLY_DEFINED`.

Exemple `.template-registration.yaml`

Dans cet exemple, la version du modèle de service est compatible avec les versions principales 1 et 2 `dumy-env-template` modèle d'environnement. Il est également compatible avec les versions majeures 1 et 3 du `another-env-template` modèle d'environnement. Le fichier ne précise pas `supported_component_sources`, les composants ne peuvent donc pas être attachés à des services basés sur cette version de modèle de service.

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

Note

Précédemment, AWS Proton défini un fichier différent, `.compatible-envs`, pour spécifier des environnements compatibles. AWS Proton supporte toujours ce fichier et son format pour des raisons de rétrocompatibilité. Nous ne recommandons plus de l'utiliser, car il n'est pas extensible et ne peut pas prendre en charge de nouvelles fonctionnalités telles que les composants.

Afficher les détails de configuration de synchronisation des modèles

Affichez les données détaillées de configuration de synchronisation des modèles à l'aide de la console ou de l'interface de ligne de commande.

AWS Management Console

Utilisez la console pour consulter les détails de configuration de synchronisation des modèles.

1. Dans le volet de navigation, choisissez **Modèles** (environnement ou service).
2. Pour afficher les données détaillées, choisissez le nom d'un modèle pour lequel vous avez créé une configuration de synchronisation de modèles.

3. Dans la page détaillée du modèle, sélectionnez **Synchronisation** onglet pour afficher les données détaillées de configuration de synchronisation du modèle.

AWS CLI

Utilisez le **AWS CLI** pour afficher un modèle synchronisé.

Exécutez la commande suivante .

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

La réponse est la suivante.

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

Utilisez le **AWS CLI** pour obtenir l'état de synchronisation des modèles.

Pour `template-version`, entrez la version principale du modèle.

Exécutez la commande suivante .

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

Modifier la configuration de synchronisation d'un modèle

Vous pouvez modifier tous les paramètres de configuration de synchronisation des modèles, sauf `template-name` et `template-type`.

Apprenez à modifier la configuration d'un modèle de synchronisation à l'aide de la console ou de l'interface de ligne de commande.

AWS Management Console

Modifiez une branche de configuration de synchronisation de modèles à l'aide de la console.

Dans la liste des modèles.

1. Dans le [AWS Proton console](#), choisissez **Modèles** (environnement ou service).
2. Dans la liste des modèles, choisissez le nom du modèle avec la configuration de synchronisation des modèles que vous souhaitez modifier.
3. Dans la page détaillée du modèle, choisissez le **Synchronisation des modèles** onglet.
4. Dans le **Détails de synchronisation des modèles** section, choisissez **Modifier**.
5. Dans le **Modifier** page, dans le **Référentiel de code source** section, pour **Succursale**, sélectionnez une branche, puis choisissez **Enregistrer la configuration**.

AWS CLI

L'exemple de commande et de réponse suivant montre comment modifier la configuration d'un modèle de synchronisation `branch` à l'aide de la CLI.

Exécutez la commande suivante .

```
$ 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"
```

La réponse est la suivante.

```
{
```

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

Vous pouvez également utiliser le **AWS CLI** pour mettre à jour les modèles de services synchronisés.

Supprimer un modèle de configuration de synchronisation

Supprimez un modèle de configuration de synchronisation à l'aide de la console ou de l'interface de ligne de commande.

AWS Management Console

Supprimez un modèle de configuration de synchronisation à l'aide de la console.

1. Sur la page de détails du modèle, sélectionnez **Synchronisation** onglet.
2. Dans le **Détails de synchronisation** section, choisissez **Déconnecter**.

AWS CLI

Les exemples de commandes et de réponses suivants montrent comment utiliser le **AWS CLI** pour supprimer les configurations de modèles synchronisées.

Exécutez la commande suivante .

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

La réponse est la suivante.

```
{
  "templateSyncConfig": {
```

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

Configurations de synchronisation des services

La synchronisation des services vous permet de configurer et de déployer vos AWS Proton services à l'aide de Git. Vous pouvez utiliser la synchronisation des services pour gérer les déploiements initiaux et les mises à jour de votre AWS Proton service à l'aide d'une configuration définie dans un référentiel Git. Grâce à Git, vous pouvez utiliser des fonctionnalités telles que le suivi des versions et les pull requests pour configurer, gérer et déployer vos services. Service Sync combine AWS Proton Git pour vous aider à fournir une infrastructure standardisée définie et gérée à l'aide AWS Proton de modèles. Il gère les définitions de service dans votre référentiel Git et réduit le nombre de changements d'outils. Par rapport à l'utilisation de Git seul, la standardisation des modèles et du déploiement vous AWS Proton permet de passer moins de temps à gérer votre infrastructure. AWS Proton offre également une transparence et une auditabilité accrues à la fois pour les développeurs et les équipes de plateforme.

AWS Proton Fichier OPS

Le `proton-ops` fichier définit l'emplacement AWS Proton du fichier de spécifications utilisé pour mettre à jour votre instance de service. Il définit également l'ordre dans lequel les instances de service doivent être mises à jour et à quel moment promouvoir les modifications d'une instance à l'autre.

Le `proton-ops` fichier prend en charge la synchronisation d'une instance de service à l'aide du fichier de spécifications, ou de plusieurs fichiers de spécifications, trouvés dans votre référentiel lié. Vous pouvez le faire en définissant un bloc de synchronisation dans le `proton-ops` fichier, comme dans l'exemple suivant.

Exemple. `/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
```

Dans l'exemple précédent, `frontend-svc` est le nom du service et `alpha`, `gamma`, `prod-one`, `prod-two`, et `prod-three` sont les instances.

Le spec fichier peut contenir toutes les instances ou un sous-ensemble des instances définies dans le `proton-ops` fichier. Cependant, au minimum, l'instance doit être définie dans la branche et la spécification à partir de laquelle elle est synchronisée. Si les instances ne sont pas définies dans le `proton-ops` fichier, avec la branche et l'emplacement du spec fichier spécifiques, Service Sync ne créera ni ne mettra à jour ces instances.

Les exemples suivants montrent à quoi ressemblent les spec fichiers. N'oubliez pas que le `proton-ops` fichier est synchronisé à partir de ces spec fichiers.

Exemple `./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"
```

Exemple `./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"
```

Exemple `./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"
```

Si une instance ne se synchronise pas et que le problème persiste lors de la tentative de synchronisation, appeler l'[GetServiceInstanceSyncStatus](#) API peut aider à résoudre le problème.

Note

Les clients qui utilisent Service Sync sont toujours soumis à AWS Proton certaines restrictions.

Bloqueurs

En synchronisant votre service à l'aide de AWS Proton Service Sync, vous pouvez mettre à jour vos spécifications de service et créer et mettre à jour des instances de service à partir de votre référentiel Git. Toutefois, il peut arriver que vous deviez mettre à jour un service ou une instance manuellement via le AWS Management Console ou AWS CLI.

AWS Proton permet d'éviter de remplacer les modifications manuelles que vous apportez via le AWS Management Console ou AWS CLI, telles que la mise à jour d'une instance de service ou la suppression d'une instance de service. Pour ce faire, crée AWS Proton automatiquement un bloqueur de synchronisation des services en désactivant la synchronisation des services lorsqu'il détecte une modification manuelle.

Pour obtenir tous les bloqueurs associés à un service, vous devez procéder comme suit dans l'ordre pour chacun des bloqueurs `serviceInstance` associés au service :

- Appelez l'`getServiceSyncBlockerSummary` API avec uniquement le `serviceName`.
- Appelez l'`getServiceSyncBlockerSummary` API avec le `serviceName` et `serviceInstanceName`.

Cela renvoie la liste des bloqueurs les plus récents et l'état qui leur est associé. Si des bloqueurs sont marqués comme `ACTIFS`, vous devez les résoudre en appelant l'`updateServiceSyncBlocker` API avec le `blockerId` et `resolvedReason` pour chacun d'entre eux.

Si vous mettez à jour ou créez manuellement une instance de service, AWS Proton crée un bloqueur de synchronisation de service sur l'instance de service. AWS Proton continue de synchroniser toutes les autres instances de service, mais désactive la synchronisation de cette instance de service jusqu'à ce que le bloqueur soit résolu. Si vous supprimez une instance de service d'un service, AWS

Proton crée un bloqueur de synchronisation de service sur le service. Cela AWS Proton empêche la synchronisation des instances de service jusqu'à ce que le bloqueur soit résolu.

Une fois que vous avez tous les bloqueurs actifs, vous devez les résoudre en appelant l'`UpdateServiceSyncBlockerAPI` avec le `blockerId` et `resolvedReason` pour chacun des bloqueurs actifs.

À l'aide de l'`AWS Management Console`, vous pouvez déterminer si la synchronisation d'un service est désactivée en accédant à l'onglet Synchronisation des services AWS Proton et en le sélectionnant. Si le service ou les instances de service sont bloqués, un bouton Activer apparaît. Pour activer la synchronisation des services, choisissez Activer.

Rubriques

- [Création d'une configuration de synchronisation des services](#)
- [Affichez les détails de la configuration d'une version CEV](#)
- [Modification d'une configuration de synchronisation de service](#)
- [Supprimer une configuration de synchronisation de services](#)

Création d'une configuration de synchronisation des services

Vous pouvez créer une configuration de synchronisation de service à l'aide de la console ou de la `AWS CLI`.

AWS Management Console

1. Sur la page Choisir un modèle de service, sélectionnez un modèle et choisissez Configurer.
2. Sur la page Configurer le service, dans la section Détails du service, entrez un nouveau nom de service.
3. (Facultatif) Saisissez une description du service.
4. Dans la section Référentiel du code source de l'application, choisissez Choisir un référentiel Git lié pour sélectionner un référentiel auquel vous avez déjà créé un lien AWS Proton. Si vous n'avez pas encore de dépôt lié, choisissez Lier un autre dépôt Git et suivez les instructions de la [section Créer un lien vers votre dépôt](#).
5. Pour Repository, choisissez le nom de votre référentiel de code source dans la liste.
6. Pour Branch, choisissez le nom de la branche du référentiel pour votre code source dans la liste.

7. (Facultatif) Dans la section Balises, choisissez Ajouter une nouvelle étiquette et entrez une clé et une valeur pour créer une étiquette gérée par le client.
8. Choisissez Suivant.
9. Sur la page Configurer les instances de service, dans la section Source de définition du service, sélectionnez Synchroniser votre service depuis Git.
10. Dans la section Fichiers de définition de service, si vous AWS Proton souhaitez créer votre `proton-ops` fichier, sélectionnez Je veux qu'AWS Proton crée les fichiers. Avec cette option, AWS Proton crée le `proton-ops` fichier spec et aux emplacements que vous spécifiez. Sélectionnez Je fournis mes propres fichiers pour créer votre propre fichier OPS.
11. Dans la section Référentiel de définition de service, choisissez Choisir un référentiel Git lié pour sélectionner un référentiel auquel vous avez déjà créé un lien AWS Proton.
12. Pour Nom du référentiel, choisissez le nom de votre référentiel de code source dans la liste.
13. Pour la branche de **proton-ops** fichier, choisissez le nom de votre branche dans la liste où AWS Proton seront placés votre fichier OPS et votre fichier de spécifications.
14. Dans la section Instances de service, chaque champ est automatiquement rempli en fonction des valeurs du `proton-ops` fichier.
15. Choisissez Suivant et passez en revue vos entrées.
16. Sélectionnez Create (Créer).

AWS CLI

Créez une configuration de synchronisation des services à l'aide du AWS CLI

- Exécutez la commande suivante .

```
$ 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)
```

La réponse positive est la suivante.

```
{  
  "serviceSyncConfig": {
```

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

Affichez les détails de la configuration d'une version CEV

Vous pouvez consulter les données de configuration d'une synchronisation de service à l'aide de la console ou AWS CLI.

AWS Management Console

Utilisez la console pour afficher les détails de la configuration d'une version CEV

1. Dans le panneau de navigation, choisissez Services.
2. Pour afficher les données détaillées, choisissez le nom du service pour lequel vous avez créé une configuration de synchronisation des services.
3. Sur la page de détails du service, sélectionnez l'onglet Synchronisation du service pour afficher les données détaillées de configuration pour la synchronisation du service.

AWS CLI

Utilisez le AWS CLI pour obtenir un service synchronisé.

Exécutez la commande suivante .

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

La réponse positive est la suivante.

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

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

Utilisez le AWS CLI pour obtenir l'état de synchronisation du service.

Exécutez la commande suivante .

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

Modification d'une configuration de synchronisation de service

Vous pouvez modifier une configuration de synchronisation de service à l'aide de la console ou de laAWS CLI.

AWS Management Console

Modifiez une configuration de synchronisation de service à l'aide de la console.

1. Dans le panneau de navigation, choisissez Services.
2. Pour afficher les données détaillées, choisissez le nom du service pour lequel vous avez créé une configuration de synchronisation des services.
3. Sur la page détaillée du service, choisissez l'onglet Synchronisation des services.
4. Dans la section Synchronisation des services, choisissez Modifier.
5. Sur la page Modifier, mettez à jour les informations que vous souhaitez modifier, puis choisissez Enregistrer.

AWS CLI

L'exemple de commande et de réponse suivant montre comment modifier une configuration de synchronisation de service à l'aide duAWS CLI.

Exécutez la commande suivante .

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

```
--service-name "service name" \  
--repository-provider "GITHUB" \  
--repository "example/proton-sync-service" \  
--ops-file-branch "main" \  
--ops-file "./configuration/custom-proton-ops.yaml"
```

La réponse positive est la suivante.

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

Supprimer une configuration de synchronisation de services

Vous pouvez supprimer une configuration de synchronisation de service à l'aide de la console ou de la AWS CLI.

AWS Management Console

Suppression d'une configuration de synchronisation de service à l'aide de la console

1. Dans la page des détails du service, choisissez l'onglet Synchronisation des services.
2. Dans la section Détails de synchronisation des services, choisissez Déconnecter pour déconnecter votre référentiel. Une fois votre référentiel déconnecté, nous ne synchronisons plus le service à partir de ce référentiel.

AWS CLI

Les exemples de commandes et de réponses suivants montrent comment utiliser le AWS CLI pour supprimer les configurations synchronisées du service.

Exécutez la commande suivante .


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

```

```
--service-name "service name"
```

La réponse positive est la suivante.

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

 Note

La synchronisation des services ne supprime pas les instances de service. Elle supprime uniquement la configuration.

Environnements AWS Proton

Pour AWS Proton, un environnement représente l'ensemble de ressources et de politiques partagées qui AWS Proton [services](#) sont déployés dans. Ils peuvent contenir toutes les ressources susceptibles d'être partagées entre AWS Proton instances de service. Ces ressources peuvent inclure des VPC, des clusters, des équilibreurs de charge partagés ou des passerelles d'API. Un AWS Proton l'environnement doit être créé avant qu'un service puisse y être déployé.

Cette section explique comment gérer les environnements à l'aide d'opérations de création, d'affichage, de mise à jour et de suppression. Pour plus d'informations, consultez le [Le AWS Proton Référence de l'API de service](#).

Rubriques

- [Rôles IAM](#)
- [Création d'un environnement](#)
- [Afficher les données d'environnement](#)
- [Mettre à jour un environnement](#)
- [Supprimer un environnement](#)
- [Connexions aux comptes d'environnement](#)
- [Environnements gérés par le client](#)
- [CodeBuild création d'un rôle de provisionnement](#)

Rôles IAM

Avec AWS Proton, vous fournissez les rôles et les AWS KMS clés IAM pour les AWS ressources que vous possédez et gérez. Ils sont ensuite appliqués et utilisés par les ressources détenues et gérées par les développeurs. Vous créez un rôle IAM pour contrôler l'accès de votre équipe de développeurs à l'AWS Proton API.

Rôle de service AWS Proton

Lorsque vous créez un nouvel environnement, vous fournissez un rôle de service IAM associé. Le rôle contient toutes les autorisations nécessaires pour mettre à jour toute l'infrastructure provisionnée définie à la fois dans les modèles d'environnement et les modèles de service. Pour des exemples de rôles, voir [AWS Proton rôle de service pour le provisionnement à l'aide AWS CloudFormation](#). Si vous

utilisez des connexions de comptes d'environnement et des comptes d'environnement, vous créez le rôle dans un compte d'environnement sélectionné. Pour en savoir plus, consultez les sections [Création d'un environnement sur un compte et provisionnement sur un autre compte](#) et [Connexions aux comptes d'environnement](#).

La manière dont vous fournissez ce rôle de service et la personne qui assume ce rôle dépendent de la méthode de provisionnement de votre environnement.

- **AWS-provisionnement géré** : vous attribuez le rôle AWS Proton, soit directement lors de la création d'un environnement, soit indirectement par le biais de connexions à des comptes. AWS Proton assume le rôle dans le compte pertinent pour fournir un environnement et une infrastructure de service.
- **Provisionnement autogéré** : il est de votre responsabilité de configurer l'automatisation de votre approvisionnement pour qu'elle joue un rôle approprié en utilisant les informations d'identification appropriées lorsqu'une pull request (PR) déclenche une action de provisionnement. Pour un exemple GitHub d'action qui assume un rôle, voir [Assumer un rôle](#) dans la documentation « Configurer les AWS informations d'identification » sur l'action pour GitHub les actions.

Pour plus d'informations sur les méthodes de provisionnement, veuillez consulter [the section called "Méthodes de provisionnement"](#).

Création d'un environnement

Apprenez à créer AWS Proton environnements.

Vous pouvez créer un AWS Proton environnement de l'une des deux manières suivantes :

- Créez, gérez et provisionnez un environnement standard à l'aide d'un modèle d'environnement standard. AWS Proton fournit une infrastructure pour votre environnement.
- Connectez-vous AWS Proton à une infrastructure gérée par le client à l'aide d'un modèle d'environnement géré par le client. Vous fournissez vos propres ressources partagées en dehors de AWS Proton, puis vous fournissez des sorties de provisionnement qui AWS Proton peut utiliser.

Vous pouvez choisir l'une des différentes approches de provisionnement lorsque vous créez un environnement.

- **AWS provisionnement géré**— Créez, gérez et provisionnez un environnement sur un seul compte. AWS Proton approvisionne votre environnement.

Cette méthode prend uniquement en charge CloudFormation modèles de code d'infrastructure (IaC).

- **AWS provisionnement géré vers un autre compte**— Dans un compte de gestion unique, créez et gérez un environnement qui est provisionné sur un autre compte avec des connexions à des comptes d'environnement. AWS Proton provisionne votre environnement sur l'autre compte. Pour plus d'informations, consultez [Création d'un environnement sur un compte et provisionnement sur un autre compte](#) et [Connexions aux comptes d'environnement](#).

Cette méthode prend uniquement en charge CloudFormation Modèles IaC.

- **Provisionnement autogéré**— AWS Proton soumet les pull requests de provisionnement à un référentiel lié avec votre propre infrastructure de provisionnement.

Cette méthode ne prend en charge que les modèles Terraform IaC.

- **CodeBuild provisionnement**— AWS Proton les usages AWS CodeBuild pour exécuter les commandes du shell que vous fournissez. Vos commandes peuvent lire les entrées qui AWS Proton fournit et est responsable du provisionnement ou du déprovisionnement de l'infrastructure et de la génération des valeurs de sortie. Un ensemble de modèles pour cette méthode inclut vos commandes dans un fichier manifeste et tous les programmes, scripts ou autres fichiers dont ces commandes peuvent avoir besoin.

À titre d'exemple d'utilisation CodeBuild provisionnement, vous pouvez inclure du code qui utilise AWS Cloud Development Kit (AWS CDK) à la fourniture AWS des ressources et un manifeste qui installe le CDK et exécute le code de votre CDK.

Pour plus d'informations, veuillez consulter [the section called “CodeBuild offre groupée”](#).

Note

Vous pouvez utiliser CodeBuild provisionnement avec des environnements et des services. Pour le moment, vous ne pouvez pas provisionner des composants de cette manière.

Avec AWS provisionnement géré (à la fois sur le même compte et vers un autre compte), AWS Proton passe des appels directs pour approvisionner vos ressources.

Grâce au provisionnement autogéré, AWS Proton émet des pull requests pour fournir des fichiers IaC compilés que votre moteur IaC utilise pour provisionner des ressources.

Pour plus d'informations, consultez [the section called “Méthodes de provisionnement”](#), [the section called “Packs de modèles”](#) et [the section called “Exigences relatives au schéma d'environnement”](#).

Rubriques

- [Création et mise en service d'un environnement standard sur le même compte](#)
- [Création d'un environnement sur un compte et provisionnement sur un autre compte](#)
- [Création et provisionnement d'un environnement à l'aide d'un provisionnement autogéré](#)

Création et mise en service d'un environnement standard sur le même compte

Utilisez la console ou AWS CLI pour créer et mettre en service un environnement sur un seul compte. Le provisionnement est géré par AWS.

AWS Management Console

Utiliser la console pour créer et mettre en service un environnement dans un seul compte

1. Dans le [AWS Proton console](#), choisissez **Environnements**.
2. Choisissez **Create environment**.
3. Dans le **Choisissez un modèle d'environnement** page, sélectionnez un modèle et choisissez **Configurez**.
4. Dans le **Configuration de l'environnement** page, dans **Approvisionnement** section, choisissez **AWS provisionnement géré**.
5. Dans le **Compte de déploiement** section, choisissez **Ce Compte AWS**.
6. Dans le **Configuration de l'environnement** page, dans **Paramètres de l'environnement** section, entrez une **Nom** de l'environnement.
7. (Facultatif) Entrez une description de l'environnement.
8. Dans le **Rôles environnementaux** section, sélectionnez la **AWS Proton rôle de service** que vous avez créé dans le cadre de [Configuration des rôles AWS Proton de service](#).
9. (Facultatif) Dans le **Rôle du composant** section, sélectionnez un rôle de service qui permet aux composants définis directement de s'exécuter dans l'environnement et qui limite les ressources qu'ils peuvent provisionner. Pour plus d'informations, veuillez consulter [Composants](#).

10. (Facultatif) Dans le **Étiquettes** section, choisissez **Ajouter un nouveau tag** entrez une clé et une valeur pour créer un tag géré par le client.
11. Choisissez **Suivant**.
12. Dans le **Configurer les paramètres personnalisés de l'environnement** page, vous devez entrer des valeurs pour `required` paramètres. Vous pouvez entrer des valeurs pour `optional` paramètres ou utilisez les valeurs par défaut lorsqu'elles sont données.
13. Choisissez **Suivant** et passez en revue vos entrées.
14. Sélectionnez **Create (Créer)**.

Consultez les détails et l'état de l'environnement, ainsi que **AWS** balises gérées et balises gérées par le client pour votre environnement.

15. Dans le panneau de navigation, choisissez **Environnements (Environnements)**.

Une nouvelle page affiche la liste de vos environnements ainsi que leur état et d'autres détails sur l'environnement.

AWS CLI

Utilisez le **AWS CLI** pour créer et mettre en service un environnement sur un seul compte.

Pour créer un environnement, vous devez spécifier [AWS Proton rôle de service](#) ARN, chemin d'accès à votre fichier de spécifications, nom de l'environnement, ARN du modèle d'environnement, versions principale et secondaire et description (facultatif).

Les exemples suivants montrent un **YAML** fichier de spécifications formaté qui spécifie les valeurs de deux entrées définies dans le fichier de schéma du modèle d'environnement. Vous pouvez utiliser le `get-environment-template-minor-version` commande pour afficher le schéma du modèle d'environnement.

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

Créez un environnement en exécutant la commande suivante.

```
$ aws proton create-environment \
```

```

--name "MySimpleEnv" \
--template-name simple-env \
--template-major-version 1 \
--proton-service-role-arn "arn:aws:iam::123456789012:role/AWSProtonServiceRole"
\
--spec "file://env-spec.yaml"

```

Réponse :

```

{
  "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"
  }
}

```

Après avoir créé un nouvel environnement, vous pouvez consulter la liste des AWS Set des balises gérées par le client, comme indiqué dans l'exemple de commande suivant. AWS Proton génère automatiquement AWS balises gérées pour vous. Vous pouvez également modifier et créer des balises gérées par le client à l'aide du AWS CLI. Pour plus d'informations, veuillez consulter [AWS Proton ressources et balises](#).

Commande :

```

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

```

Création d'un environnement sur un compte et provisionnement sur un autre compte

Utilisez la console ou AWS CLI pour créer un environnement standard dans un compte de gestion qui approvisionne l'infrastructure de l'environnement dans un autre compte. Le provisionnement est géré par AWS.

Avant d'utiliser la console ou l'interface de ligne de commande, procédez comme suit.

1. Identifiez le `Compte AWSID` du compte de gestion et d'environnement, et copiez-les pour une utilisation ultérieure.
2. Dans le compte d'environnement, créez un `AWS Proton` rôle de service avec des autorisations minimales pour que l'environnement puisse créer. Pour plus d'informations, veuillez consulter [AWS Proton rôle de service pour le provisionnement à l'aide AWS CloudFormation](#).

AWS Management Console

Utilisez la console pour créer un environnement dans un compte et le provisionner dans un autre.


1. Dans le compte d'environnement, créez une connexion au compte d'environnement et utilisez-la pour envoyer une demande de connexion au compte de gestion.
 - a. Dans [AWS Proton console](#), choisissez `Connexions aux comptes d'environnement` dans le volet de navigation.
 - b. Dans le `Connexions aux comptes d'environnement` page, choisissez `Demande de connexion`.

Note

Vérifiez que l'identifiant du compte qui figure dans `Connexion au compte d'environnement` l'en-tête de page correspond à l'identifiant de votre compte d'environnement pré-identifié.

- c. Dans le `Demande de connexion` page, dans `Rôle environnemental` section, sélectionnez `Rôle de service existant` et le nom du rôle de service que vous avez créé pour l'environnement.
- d. Dans le `Connectez-vous au compte de gestion` section, entrez le `ID` du compte de gestion et un `Nom de l'environnement` pour votre `AWS Proton` environnement. Copiez le nom pour une utilisation ultérieure.
- e. Choisissez `Demande de connexion` dans le coin inférieur droit de la page.
- f. Votre demande apparaît comme étant en attente dans le `Connexions d'environnement envoyées à un compte de gestion` Un tableau et une fenêtre modale indiquent comment accepter la demande depuis le compte de gestion.

2. Dans le compte de gestion, acceptez une demande de connexion depuis le compte d'environnement.
 - a. Connectez-vous à votre compte de gestion et choisissez Connexions aux comptes d'environnement dans le AWS Proton console.
 - b. Dans le Connexions aux comptes d'environnement page, dans Demandes de connexion à un compte d'environnement tableau, sélectionnez la connexion au compte d'environnement avec l'ID de compte d'environnement qui correspond à votre ID de compte d'environnement pré-identifié.

 Note

Vérifiez que l'identifiant du compte qui figure dans Connexion au compte d'environnement l'en-tête de page correspond à l'identifiant de votre compte de gestion pré-identifié.

- c. Choisissez Accepter. Le statut passe de EN ATTENTE à CONNECTÉ.
3. Dans le compte de gestion, créez un environnement.
 - a. Dans le volet de navigation, choisissez Modèles d'environnement.
 - b. Dans le Modèles d'environnement page, choisissez Création d'un modèle d'environnement.
 - c. Dans le Choisissez un modèle d'environnement page, choisissez un modèle d'environnement.
 - d. Dans le Configuration de l'environnement page, dans Approvisionnement section, choisissez AWS provisionnement géré.
 - e. Dans le Compte de déploiement section, choisissez Un autre AWS compte ;.
 - f. Dans le Détails de l'environnement section, sélectionnez votre Connexion au compte d'environnement et Nom de l'environnement.
 - g. Choisissez Suivant.
 - h. Remplissez les formulaires et choisissez Suivant jusqu'à ce que vous atteigniez Révision et création page.
 - i. Vérifiez et choisissez Créer un environnement.

AWS CLI

Utilisez le **AWS CLI** pour créer un environnement dans un compte et le provisionner dans un autre.

Dans le compte d'environnement, créez une connexion au compte d'environnement et demandez à vous connecter en exécutant la commande suivante.

```
$ 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"
```

Réponse :

```
{  
  "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"  
  }  
}
```

Dans le compte de gestion, acceptez la demande de connexion au compte d'environnement en exécutant la commande suivante.

```
$ aws proton accept-environment-account-connection \  
  --id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

Réponse :

```
{
```

```

"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"
}
}

```

Consultez la connexion à votre compte d'environnement en exécutant la commande suivante.

```

$ aws proton get-environment-account-connection \
  --id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"

```

Réponse :

```

{
  "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"
  }
}

```

Dans le compte de gestion, créez un environnement en exécutant la commande suivante.

```

$ 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"
```

Réponse :

```
{  
  "environment": {  
    "arn": "arn:aws:proton:region-id:111111111111:environment/simple-env-  
connected",  
    "createdAt": "2021-04-28T23:02:57.944000+00:00",  
    "deploymentStatus": "IN_PROGRESS",  
    "environmentAccountId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
    "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"  
  }  
}
```

Création et provisionnement d'un environnement à l'aide d'un provisionnement autogéré

Lorsque vous utilisez le provisionnement autogéré, AWS Proton soumet les pull requests de provisionnement à un référentiel lié avec votre propre infrastructure de provisionnement. Les pull requests démarrent votre propre flux de travail, qui appelle AWS services ; pour fournir des infrastructures.

Considérations relatives au provisionnement autogéré :

- Avant de créer un environnement, configurez un répertoire de ressources du référentiel pour le provisionnement autogéré. Pour plus d'informations, veuillez consulter [AWS Proton infrastructure sous forme de fichiers de code](#).
- Après avoir créé l'environnement, AWS Proton attend de recevoir des notifications asynchrones concernant l'état du provisionnement de votre infrastructure. Votre code d'approvisionnement doit utiliser le AWS Proton `NotifyResourceStateChangeAPI` permettant d'envoyer ces notifications asynchrones à AWS Proton.

Vous pouvez utiliser le provisionnement autogéré dans la console ou avec AWS CLI. Les exemples suivants montrent comment utiliser le provisionnement autogéré avec Terraform.

AWS Management Console

Utilisez la console pour créer un environnement Terraform à l'aide du provisionnement autogéré.

1. Dans le [AWS Proton console](#), choisissez Environnements.
2. Choisissez Create environment.
3. Dans le Choisissez un modèle d'environnement page, sélectionnez un modèle Terraform et choisissez Configurez.
4. Dans le Configuration de l'environnement page, dans Approvisionnement section, choisissez Provisionnement autogéré.
5. Dans le Détails du référentiel de provisionnement section :
 - a. Si vous ne l'avez pas encore fait [lié votre référentiel de provisionnement à AWS Proton](#), choisissez Nouveau référentiel, choisissez l'un des fournisseurs de référentiels, puis, pour CodeStar connexion, choisissez l'une de vos connexions.

Note

Si vous n'êtes pas encore connecté au compte du fournisseur de référentiel concerné, choisissez Ajouter un nouveau CodeStar connexion. Créez ensuite une connexion, puis cliquez sur le bouton d'actualisation situé à côté du CodeStar connexion menu. Vous devriez maintenant pouvoir choisir votre nouvelle connexion dans le menu.

Si vous avez déjà lié votre référentiel à AWS Proton, choisissez Référentiel existant.

- b. Pour Nom du référentiel, choisissez un référentiel. Le menu déroulant affiche les référentiels liés pour Référentiel existant ou la liste des référentiels du compte fournisseur pour Nouveau référentiel.
 - c. Pour Nom de la succursale, choisissez l'une des branches du référentiel.
6. Dans le Paramètres de l'environnement section, entrez un Nom de l'environnement.
 7. (Facultatif) Entrez une description de l'environnement.

8. (Facultatif) Dans le **Étiquettes** section, choisissez **Ajouter un nouveau tag** entrez une clé et une valeur pour créer un tag géré par le client.
9. Choisissez **Suivant**.
10. Dans le **Configurer les paramètres personnalisés de l'environnement** page, vous devez entrer des valeurs pour **required** paramètres. Vous pouvez entrer des valeurs pour **optional** paramètres ou utilisez les valeurs par défaut lorsqu'elles sont données.
11. Choisissez **Suivant** et passez en revue vos entrées.
12. Choisissez **Créer** pour envoyer une pull request.
 - Si vous approuvez la pull request, le déploiement est en cours.
 - Si vous rejetez la pull request, la création de l'environnement est annulée.
 - Si le délai d'expiration de la pull request est dépassé, création de l'environnement n'est pas complet.
13. Consultez les détails et l'état de l'environnement, ainsi que **AWS** balises gérées et balises gérées par le client pour votre environnement.
14. Dans le panneau de navigation, choisissez **Environnements (Environnements)**.

Une nouvelle page affiche la liste de vos environnements ainsi que leur état et d'autres détails sur l'environnement.

AWS CLI

Lorsque vous créez un environnement à l'aide d'un provisionnement autogéré, vous ajoutez le `provisioningRepository` paramètre et omettez le `ProtonServiceRoleArn` et `environmentAccountConnectionId` paramètres.

Utilisez le **AWS CLI** pour créer un environnement Terraform avec un provisionnement autogéré.

1. Créez un environnement et envoyez une pull request au référentiel pour révision et approbation.

Les exemples suivants montrent un **YAML** fichier de spécifications formaté qui définit les valeurs de deux entrées en fonction du fichier de schéma du modèle d'environnement. Vous pouvez utiliser le `get-environment-template-minor-version` commande pour afficher le schéma du modèle d'environnement.

Spécification :

```
proton: EnvironmentSpec
spec:
  ssm_parameter_value: "test"
```

Créez un environnement en exécutant la commande suivante.

```
$ 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"
```

Réponse : >

```
{
  "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. Vérifiez la demande.

- Si vous approuvez la demande, le provisionnement est en cours.
- Si vous rejetez la demande, la création de l'environnement est annulée.
- Si le délai d'expiration de la pull request est dépassé, création de l'environnement n'est pas complet.

3. Fournissez de manière asynchrone l'état de provisionnement à AWS Proton. L'exemple suivant indique AWS Proton d'un approvisionnement réussi.

```
$ aws proton notify-resource-deployment-status-change \
  --resource-arn "arn:aws:proton:region-id:123456789012:environment/pr-
  environment" \
  --status "SUCCEEDED"
```

Afficher les données d'environnement

Vous pouvez consulter les données détaillées de l'environnement à l'aide de AWS Proton console ou AWS CLI.

AWS Management Console

Vous pouvez consulter des listes d'environnements avec des détails et des environnements individuels avec des données détaillées en utilisant [AWS Proton console](#).

1. Pour afficher la liste de vos environnements, choisissez Environnements dans le volet de navigation.
2. Pour afficher les données détaillées, choisissez le nom d'un environnement.

Consultez les données détaillées de votre environnement.

AWS CLI

Utilisez le AWS CLI obtenir ou lister des détails de l'environnement.

Exécutez la commande suivante :

```
$ aws proton get-environment \
  --name "MySimpleEnv"
```

Réponse :

```
{
  "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"
  }
}

```

Mettre à jour un environnement

Si le `protonServiceRoleArn` paramètre est associé à une connexion à un compte d'environnement, ne mettre à jour ou inclure le `protonServiceRoleArn` paramètre pour mettre à jour ou se connecter à une connexion à un compte d'environnement.

Vous ne pouvez effectuer la mise à jour vers une nouvelle connexion au compte d'environnement que si les deux conditions suivantes sont remplies :

- La connexion au compte d'environnement a été créée dans le même compte d'environnement que celui dans lequel la connexion au compte d'environnement actuel a été créée.
- >La connexion au compte d'environnement est associée à l'environnement actuel.

Si l'environnement n'est pas associé à une connexion à un compte d'environnement, ne mettre à jour ou inclure le `environmentAccountConnectionId` paramètre.

Vous pouvez mettre à jour l'un ou l'autre `environmentAccountConnectionId` ou `protonServiceRoleArn` paramètre et valeur. Vous ne pouvez pas mettre à jour les deux.

Si votre environnement utilise un provisionnement autogéré, ne mettre à jour le `provisioning-repository` paramètre et mettre le `environmentAccountConnectionId` et `protonServiceRoleArn` paramètres.

Il existe quatre modes de mise à jour d'un environnement, comme décrit dans la liste suivante. Lorsque vous utilisez le `AWS CLI`, le `deployment-type` champ définit le mode. Lorsque vous

utilisez la console, ces modes correspondent au `Modifier`, `Mettre à jour`, `Mise à jour mineure`, et `Mise à jour majeure` actions qui s'affichent depuis `Actions`.

NONE

Dans ce mode, un déploiement ne se produit. Seuls les paramètres de métadonnées demandés sont mis à jour.

CURRENT_VERSION

Dans ce mode, l'environnement est déployé et mis à jour avec les nouvelles spécifications que vous fournissez. Seuls les paramètres demandés sont mis à jour. Ne le faites pas inclure des paramètres de version mineurs ou majeurs lorsque vous utilisez ce `deployment-type`.

MINOR_VERSION

Dans ce mode, l'environnement est déployé et mis à jour avec la version mineure (la plus récente) publiée et recommandée de la version majeure actuellement utilisée par défaut. Vous pouvez également spécifier une version secondaire différente de la version principale actuellement utilisée.

MAJOR_VERSION

Dans ce mode, l'environnement est déployé et mis à jour avec la version majeure et mineure publiée et recommandée (la plus récente) du modèle actuel par défaut. Vous pouvez également spécifier une version principale différente qui est supérieure à la version principale utilisée et une version secondaire (facultatif).

Rubriques

- [Mettre à jour un AWS Environnement de provisionnement géré](#)
- [Mettre à jour un environnement de provisionnement autogéré](#)
- [Annuler un déploiement d'environnement en cours](#)

Mettre à jour un AWS Environnement de provisionnement géré

Le provisionnement standard n'est pris en charge que par les environnements qui fournissent avec AWS CloudFormation.

Utilisez la console ou AWS CLI pour mettre à jour votre environnement.

AWS Management Console

Mettez à jour un environnement à l'aide de la console comme indiqué dans les étapes suivantes.

1. Choisissez l'une des deux étapes suivantes.
 - a. Dans la liste des environnements.
 - i. Dans le [AWS Proton console](#), choisissez Environnements.
 - ii. Dans la liste des environnements, cliquez sur le bouton radio situé à gauche de l'environnement que vous souhaitez mettre à jour.
 - b. Sur la page détaillée de l'environnement de la console.
 - i. Dans le [AWS Proton console](#), choisissez Environnements.
 - ii. Dans la liste des environnements, choisissez le nom de l'environnement que vous souhaitez mettre à jour.
2. Choisissez l'une des 4 étapes suivantes pour mettre à jour votre environnement.
 - a. Pour effectuer une modification qui ne nécessite pas le déploiement de l'environnement.
 - i. Par exemple, pour modifier une description.

Choisissez Edit (Modifier).
 - ii. Remplissez le formulaire et choisissez Suivant.
 - iii. Vérifiez votre modification et choisissez Mettre à jour.
 - b. Pour mettre à jour uniquement les entrées de métadonnées.
 - i. Choisissez Actions puis Mettre à jour.
 - ii. Remplissez le formulaire et choisissez Modifier.
 - iii. Remplissez les formulaires et choisissez Suivant jusqu'à ce que vous atteigniez Révision page.
 - iv. Vérifiez vos mises à jour et choisissez Mettre à jour.
 - c. Pour mettre à jour une nouvelle version mineure de son modèle d'environnement.
 - i. Choisissez Actions puis Mise à jour mineure.
 - ii. Remplissez le formulaire et choisissez Suivant.

- iii. Remplissez les formulaires et choisissez **Suivant** jusqu'à ce que vous atteigniez **Révision** page.
 - iv. Vérifiez vos mises à jour et choisissez **Mettre à jour**.
- d. Pour effectuer une mise à jour vers une nouvelle version majeure de son modèle d'environnement.
- i. Choisissez **Actions** puis **Mise à jour majeure**.
 - ii. Remplissez le formulaire et choisissez **Suivant**.
 - iii. Remplissez les formulaires et choisissez **Suivant** jusqu'à ce que vous atteigniez **Révision** page.
 - iv. Vérifiez vos mises à jour et choisissez **Mettre à jour**.

AWS CLI

Utilisez le **AWS Proton AWS CLI** pour mettre à jour un environnement vers une nouvelle version mineure.

Exécutez la commande suivante pour mettre à jour votre environnement :

```
$ 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"
```

Réponse :

```
{  
  "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"
  }
}

```

Exécutez la commande suivante pour obtenir et confirmer l'état :

```

$ aws proton get-environment \
  --name "MySimpleEnv"

```

Réponse :

```

{
  "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"
  }
}

```

Mettre à jour un environnement de provisionnement autogéré

Le provisionnement autogéré n'est pris en charge que par les environnements qui approvisionnent avec Terraform.

Utilisez la console ou AWS CLI pour mettre à jour votre environnement.

AWS Management Console

Mettez à jour un environnement à l'aide de la console comme indiqué dans les étapes suivantes.

1. Choisissez l'une des deux étapes suivantes.
 - a. Dans la liste des environnements.
 - i. Dans le [AWS Proton console](#), choisissez Environnements.
 - ii. Dans la liste des environnements, cliquez sur le bouton radio situé à gauche du modèle d'environnement que vous souhaitez mettre à jour.
 - b. Sur la page détaillée de l'environnement de la console.
 - i. Dans le [AWS Proton console](#), choisissez Environnements.
 - ii. Dans la liste des environnements, choisissez le nom de l'environnement que vous souhaitez mettre à jour.
2. Choisissez l'une des 4 étapes suivantes pour mettre à jour votre environnement.
 - a. Pour effectuer une modification qui ne nécessite pas le déploiement de l'environnement.
 - i. Par exemple, pour modifier une description.

Choisissez Edit (Modifier).
 - ii. Remplissez le formulaire et choisissez Suivant.
 - iii. Vérifiez votre modification et choisissez Mettre à jour.
 - b. Pour mettre à jour uniquement les entrées de métadonnées.
 - i. Choisissez Actions puis Mettre à jour.
 - ii. Remplissez le formulaire et choisissez Modifier.
 - iii. Remplissez les formulaires et choisissez Suivant jusqu'à ce que vous atteigniez Révision page.
 - iv. Vérifiez vos mises à jour et choisissez Mettre à jour.
 - c. Pour mettre à jour une nouvelle version mineure de son modèle d'environnement.
 - i. Choisissez Actions puis Mise à jour mineure.
 - ii. Remplissez le formulaire et choisissez Suivant.

- iii. Remplissez les formulaires et choisissez **Suivant** jusqu'à ce que vous atteigniez **Révision** page.
 - iv. Vérifiez vos mises à jour et choisissez **Mettre à jour**.
- d. Pour effectuer une mise à jour vers une nouvelle version majeure de son modèle d'environnement.
- i. Choisissez **Actions** puis **Mise à jour majeure**.
 - ii. Remplissez le formulaire et choisissez **Suivant**.
 - iii. Remplissez les formulaires et choisissez **Suivant** jusqu'à ce que vous atteigniez **Révision** page.
 - iv. Vérifiez vos mises à jour et choisissez **Mettre à jour**.

AWS CLI

Utilisez le **AWS CLI** pour mettre à jour un environnement Terraform vers une nouvelle version mineure avec un provisionnement autogéré.

1. Exécutez la commande suivante pour mettre à jour votre environnement :

```
$ 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"
```

Réponse :

```
{
  "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. Exécutez la commande suivante pour obtenir et confirmer l'état :

```

$ aws proton get-environment \
  --name pr-environment

```

Réponse :

```

{
  "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\n ssm_another_parameter_value: \"update\"\n\n",
    "templateMajorVersion": "1",
    "templateMinorVersion": "1",
    "templateName": "pr-env-template"
  }
}

```

```
}
```

3. Vérifiez la demande d'extraction envoyée par AWS Proton.
 - Si vous approuvez la demande, le provisionnement est en cours.
 - Si vous rejetez la demande, la création de l'environnement est annulée.
 - Si le délai d'expiration de la pull request est dépassé, la création de l'environnement n'est pas terminée.
4. Fournir l'état de provisionnement à AWS Proton.

```
$ aws proton notify-resource-deployment-status-change \
  --resource-arn "arn:aws:proton:region-id:123456789012:environment/pr-
  environment" \
  --status "SUCCEEDED"
```

Annuler un déploiement d'environnement en cours

Vous pouvez tenter d'annuler le déploiement d'une mise à jour d'environnement si `deploymentStatus` est dans `IN_PROGRESS`. AWS Proton tente d'annuler le déploiement. L'annulation réussie n'est pas garantie.

Lorsque vous annulez un déploiement de mise à jour, AWS Proton tente d'annuler le déploiement comme indiqué dans les étapes suivantes.

Avec l'approvisionnement géré par AWS Proton, AWS Proton effectue les opérations suivantes :

- Définit l'état du déploiement sur `CANCELLING`.
- Arrête le déploiement en cours et supprime toutes les nouvelles ressources créées par le déploiement lorsque `IN_PROGRESS`.
- Définit l'état du déploiement sur `CANCELLED`.
- Rétablit l'état de la ressource à ce qu'il était avant le début du déploiement.

Grâce au provisionnement autogéré, AWS Proton effectue les opérations suivantes :

- Tente de fermer la pull request pour empêcher la fusion des modifications apportées à votre référentiel.
- Définit l'état du déploiement sur `CANCELLED` si la pull request a été fermée avec succès.

Pour obtenir des instructions sur la façon d'annuler un déploiement d'environnement, voir [CancelEnvironmentDeployment](#) dans le [AWS Proton Référence d'API](#).

Vous pouvez utiliser la console ou l'interface de ligne de commande pour annuler les environnements en cours.

AWS Management Console

Utilisez la console pour annuler le déploiement d'une mise à jour d'environnement, comme indiqué dans les étapes suivantes.

1. Dans le [AWS Proton console](#), choisissez **Environnements** dans le volet de navigation.
2. Dans la liste des environnements, choisissez le nom de l'environnement contenant la mise à jour de déploiement que vous souhaitez annuler.
3. Si l'état de votre déploiement de mises à jour est **En cours**, sur la page détaillée de l'environnement, choisissez **Actions** puis **Annuler le déploiement**.
4. Une fenêtre modale vous invite à confirmer que vous souhaitez annuler. Choisissez **Annuler le déploiement**.
5. L'état de votre déploiement de mises à jour est défini sur **Annulation** puis **Annulé** pour terminer l'annulation.

AWS CLI

Utilisez le [AWS Proton AWS CLI](#) pour annuler un déploiement de mise à jour de l'environnement `IN_PROGRESS` vers une nouvelle version mineure 2.

Une condition d'attente est incluse dans le modèle utilisé pour cet exemple afin que l'annulation commence avant la réussite du déploiement de la mise à jour.

Exécutez la commande suivante pour annuler la mise à jour :

```
$ aws proton cancel-environment-deployment \  
    --environment-name "MySimpleEnv"
```

Réponse :

```
{
```

```

    "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"
    }
  }
}

```

Exécutez la commande suivante pour obtenir et confirmer l'état : »

```

$ aws proton get-environment \
  --name "MySimpleEnv"

```

Réponse :

```

{
  "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"
  }
}

```

Supprimer un environnement

Vous pouvez supprimer un AWS Proton environnement en utilisant AWS Proton console ou AWS CLI.

Note

Vous ne pouvez pas supprimer un environnement auquel un composant est associé. Pour supprimer un tel environnement, vous devez d'abord supprimer tous les composants qui s'exécutent dans l'environnement. Pour plus d'informations sur les composants, voir [Composants](#).

AWS Management Console

Supprimez un environnement à l'aide de la console comme décrit dans les deux options suivantes.

Dans la liste des environnements.

1. Dans le [AWS Proton console](#), choisissez Environnements.
2. Dans la liste des environnements, sélectionnez le bouton radio situé à gauche de l'environnement que vous souhaitez supprimer.
3. Choisissez Actions puis Supprimer.
4. Une fenêtre modale vous invite à confirmer l'action de suppression.
5. Suivez les instructions et choisissez Oui, supprimer.

Sur la page détaillée de l'environnement.

1. Dans le [AWS Proton console](#), choisissez Environnements.
2. Dans la liste des environnements, choisissez le nom de l'environnement que vous souhaitez supprimer.
3. Sur la page détaillée de l'environnement, choisissez Actions puis Supprimer.
4. Une fenêtre modale vous invite à confirmer que vous souhaitez supprimer.
5. Suivez les instructions et choisissez Oui, supprimer.

AWS CLI

Utilisez le **AWS CLI** pour supprimer un environnement.

Ne le faites pas supprimer un environnement si des services ou des instances de service sont déployés dans l'environnement.

Exécutez la commande suivante :

```
$ aws proton delete-environment \  
  --name "MySimpleEnv"
```

Réponse :

```
{  
  "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"  
  }  
}
```

Connexions aux comptes d'environnement

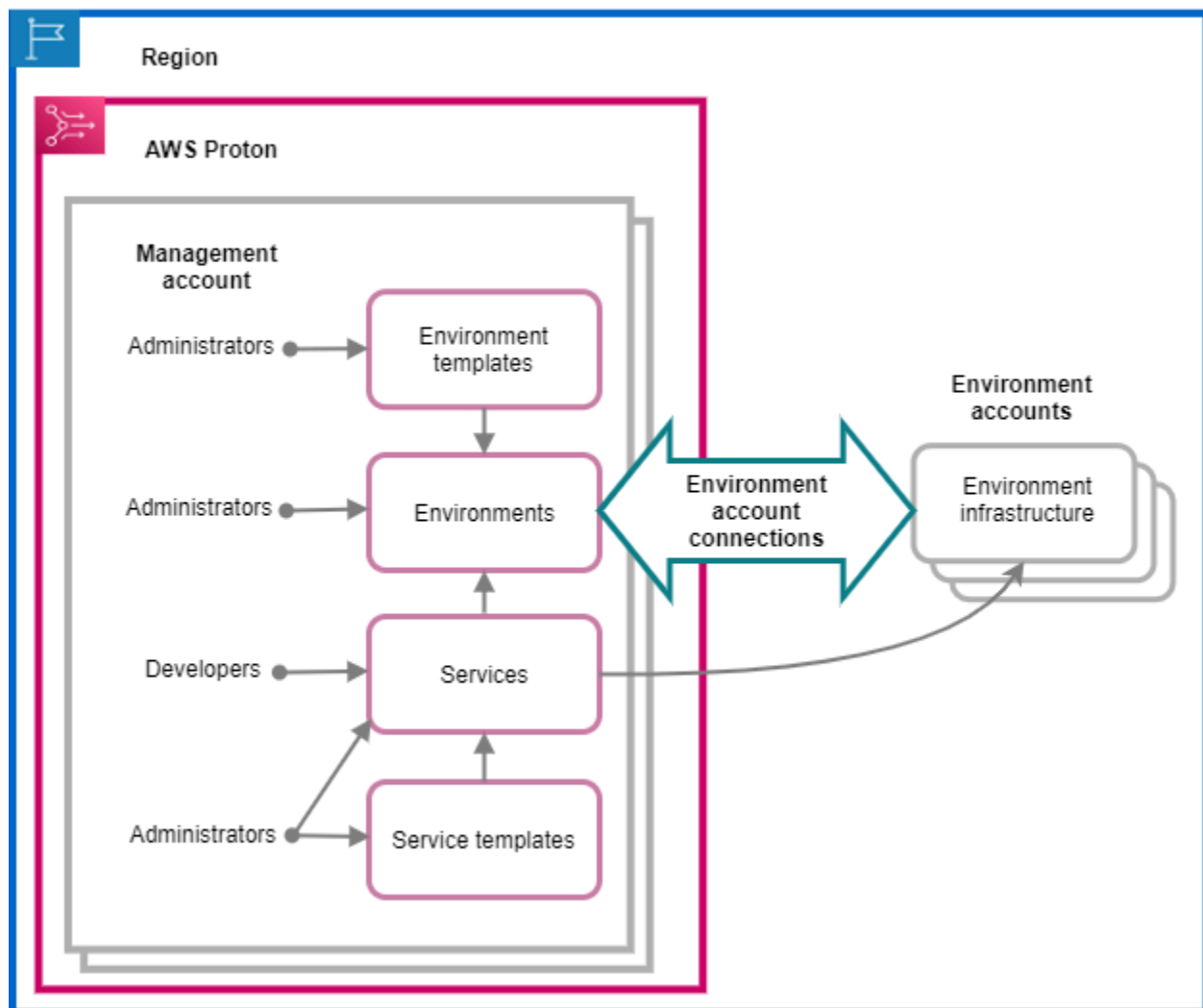
Présentation

Découvrez comment créer et gérer un **AWS Proton** environnement sur un compte et provisionnement de ses ressources d'infrastructure sur un autre compte. Cela peut contribuer à améliorer la visibilité et l'efficacité à grande échelle. Les connexions aux comptes d'environnement prennent uniquement en charge le provisionnement standard avec **AWS CloudFormation** infrastructure sous forme de code.

Note

Les informations de cette rubrique concernent les environnements configurés avec AWS Provisionnement géré. Avec des environnements configurés avec provisionnement autogéré, AWS Proton ne provisionne pas directement votre infrastructure. Au lieu de cela, il envoie des pull requests (PR) à votre référentiel pour le provisionnement. Il est de votre responsabilité de vous assurer que votre code d'automatisation adopte la bonne identité et le bon rôle.

Pour plus d'informations sur les méthodes de provisionnement, voir [the section called "Méthodes de provisionnement"](#).

Terminologie

Avec AWS Proton connexions aux comptes d'environnement, vous pouvez créer un AWS Proton environnement à partir d'un compte et provisionnement de son infrastructure sur un autre compte.

Compte de gestion

Le compte unique dans lequel vous, en tant qu'administrateur, créez un AWS Proton environnement qui fournit des ressources d'infrastructure dans un autre compte d'environnement.

Compte environnemental

Un compte dans lequel l'infrastructure de l'environnement est provisionnée lorsque vous créez un AWS Proton environnement dans un autre compte.

Connexion au compte d'environnement

Une connexion bidirectionnelle sécurisée entre un compte de gestion et un compte d'environnement. Il conserve les autorisations et les autorisations comme décrit plus en détail dans les sections suivantes.

Lorsque vous créez une connexion à un compte d'environnement dans un compte d'environnement d'une région spécifique, seuls les comptes de gestion de la même région peuvent voir et utiliser la connexion au compte d'environnement. Cela signifie que le AWS Proton l'environnement créé dans le compte de gestion et l'infrastructure d'environnement provisionnée dans le compte d'environnement doivent se trouver dans la même région.

Considérations concernant la connexion au compte d'environnement

- Vous avez besoin d'une connexion à un compte d'environnement pour chaque environnement que vous souhaitez provisionner dans un compte d'environnement.
- Pour plus d'informations sur les quotas de connexion aux comptes d'environnement, voir [Quotas AWS Proton](#).

Balisage

Dans le compte d'environnement, utilisez la console ou AWS CLI pour afficher et gérer les balises gérées par le client concernant la connexion au compte d'environnement. AWS balises gérées ne sont pas générées pour les connexions aux comptes d'environnement. Pour plus d'informations, veuillez consulter [Identification](#).

Création d'un environnement sur un compte et provisionnement de son infrastructure sur un autre compte

Pour créer et mettre en service un environnement à partir d'un compte de gestion unique, configurez un compte d'environnement pour l'environnement que vous envisagez de créer.

Démarrez dans le compte d'environnement et créez une connexion.

Dans le compte d'environnement, créez un rôle de service AWS Proton limité aux seules autorisations nécessaires au provisionnement des ressources d'infrastructure de votre environnement. Pour plus d'informations, veuillez consulter [AWS Proton rôle de service pour le provisionnement à l'aide AWS CloudFormation](#).

Créez ensuite une demande de connexion à un compte d'environnement et envoyez-la à votre compte de gestion. Lorsque la demande est acceptée, AWS Proton peut utiliser le rôle IAM associé qui permet le provisionnement des ressources d'environnement dans le compte d'environnement associé.

Dans le compte de gestion, acceptez ou refusez la connexion au compte d'environnement.

Dans le compte de gestion, acceptez ou rejetez la demande de connexion au compte d'environnement. Vous ne pouvez pas supprimer une connexion à un compte d'environnement depuis votre compte de gestion.

Si vous acceptez la demande, AWS Proton peut utiliser le rôle IAM associé qui permet le provisionnement des ressources dans le compte d'environnement associé.

Les ressources de l'infrastructure d'environnement sont provisionnées dans le compte d'environnement associé. Vous pouvez uniquement utiliser AWS Proton Des API permettant d'accéder à votre environnement et à ses ressources d'infrastructure et de les gérer à partir de votre compte de gestion. Pour plus d'informations, consultez [Création d'un environnement sur un compte et provisionnement sur un autre compte](#) et [Mettre à jour un environnement](#).

Après avoir rejeté une demande, vous ne pouvez pas accepter ou utiliser la connexion au compte d'environnement refusée.

Note

Vous ne pouvez pas rejeter la connexion d'un compte d'environnement connecté à un environnement. Pour refuser la connexion au compte d'environnement, vous devez d'abord supprimer l'environnement associé.

Dans le compte d'environnement, accédez aux ressources d'infrastructure provisionnées.

Dans le compte d'environnement, vous pouvez consulter et accéder aux ressources d'infrastructure provisionnées. Par exemple, vous pouvez utiliser `CloudFormationActions` d'API pour surveiller et nettoyer les piles si nécessaire. Vous ne pouvez pas utiliser le `AWS ProtonActions` d'API permettant d'accéder à ou de gérer `AWS Proton` environnement qui a été utilisé pour fournir les ressources d'infrastructure.

Dans le compte d'environnement, vous pouvez supprimer les connexions au compte d'environnement que vous avez créées dans le compte d'environnement. Vous ne pouvez pas les accepter ou les rejeter. Si vous supprimez une connexion à un compte d'environnement qui est utilisée par `AWS Proton` environnement, `AWS Proton` ne sera pas en mesure de gérer les ressources de l'infrastructure de l'environnement tant qu'une nouvelle connexion à l'environnement ne sera pas acceptée pour le compte d'environnement et l'environnement nommé. Vous êtes responsable du nettoyage des ressources provisionnées qui ne sont toujours pas connectées à l'environnement.

Utiliser la console ou l'interface de ligne de commande pour gérer les connexions aux comptes d'environnement


Vous pouvez utiliser la console ou l'interface de ligne de commande pour créer et gérer les connexions aux comptes d'environnement.

AWS Management Console

Utilisez la console pour créer une connexion au compte d'environnement et envoyer une demande au compte de gestion comme indiqué dans les étapes suivantes.


1. Choisissez le nom de l'environnement que vous envisagez de créer dans votre compte de gestion ou choisissez le nom d'un environnement existant qui nécessite une connexion à un compte d'environnement.
2. Dans un compte d'environnement, dans le [AWS Proton console](#), choisissez `Connexions aux comptes d'environnement` dans le volet de navigation.

3. Dans le **Connexions aux comptes d'environnement** page, choisissez **Demande de connexion**.

 Note

Vérifiez l'identifiant du compte qui figure dans **Connexion au compte d'environnement** titre de page. Assurez-vous qu'il correspond à l'ID du compte d'environnement dans lequel vous souhaitez que votre environnement désigné fasse le provisionnement.

4. Dans le **Demande de connexion** page :
 - a. Dans le **Connectez-vous au compte de gestion** section, entrez le **ID** du compte de gestion et le **Nom** de l'environnement que vous avez saisi à l'étape 1.
 - b. Dans le **Rôle environnemental** section, choisissez **Nouveau rôle de service** et **AWS Proton** crée automatiquement un nouveau rôle pour vous. Ou sélectionnez **Rôle de service existant** et le nom du rôle de service que vous avez créé précédemment.

 Note


Le rôle que **AWS Proton** crée automatiquement pour vous des autorisations étendues. Nous vous recommandons de limiter le rôle aux autorisations requises pour provisionner les ressources d'infrastructure de votre environnement. Pour plus d'informations, veuillez consulter [AWS Proton rôle de service pour le provisionnement à l'aide AWS CloudFormation](#).

- c. (Facultatif) Dans le **Étiquettes** section, choisissez **Ajouter un nouveau tag** pour créer un tag géré par le client pour la connexion à votre compte d'environnement.
 - d. Choisissez **Demande de connexion**.
5. Votre demande apparaît comme étant en attente dans le **Connexions d'environnement** envoyées à un compte de gestion. Un tableau et un modal vous permettent de savoir comment accepter la demande depuis le compte de gestion.

Acceptez ou refusez une demande de connexion à un compte d'environnement.

1. Dans un compte de gestion, dans le [AWS Proton console](#), choisissez **Connexions aux comptes d'environnement** dans le volet de navigation.

2. Dans le **Connexions aux comptes d'environnement** page, dans **Demandes de connexion** à un compte d'environnement table, choisissez la demande de connexion à l'environnement à accepter ou à rejeter.


 Note

Vérifiez l'identifiant du compte qui figure dans **Connexion au compte d'environnement** titre de page. Assurez-vous qu'il correspond à l'ID du compte de gestion associé à la connexion au compte d'environnement à rejeter. Après avoir refusé la connexion à ce compte d'environnement, vous ne pouvez pas accepter ou utiliser la connexion au compte d'environnement refusée.

3. Choisissez **Rejeter** ou **Accepter**.
 - Si vous avez sélectionné **Rejeter**, le statut passe de **en attente** pour **rejetés**.
 - Si vous avez sélectionné **Accepter**, le statut passe de **en attente** pour **connecté**.

Supprimez la connexion à un compte d'environnement.

1. Dans un compte d'environnement, dans le [AWS Proton console](#), choisissez **Connexions aux comptes d'environnement** dans le volet de navigation.

 Note

Vérifiez l'identifiant du compte qui figure dans **Connexion au compte d'environnement** titre de page. Assurez-vous qu'il correspond à l'ID du compte de gestion associé à la connexion au compte d'environnement à rejeter. Après avoir supprimé la connexion à ce compte d'environnement, AWS Proton ne peut pas gérer les ressources de l'infrastructure d'environnement dans le compte d'environnement. Il ne peut le gérer qu'après qu'une nouvelle connexion au compte d'environnement pour le compte d'environnement et que l'environnement nommé ait été accepté par le compte de gestion.

2. Dans le **Connexions aux comptes d'environnement** page, dans **Demandes envoyées pour se connecter au compte de gestion** section, choisissez **Supprimer**.
3. Une fenêtre modale vous invite à confirmer que vous souhaitez supprimer. Choisissez **Delete (Supprimer)**.

AWS CLI

Choisissez le nom de l'environnement que vous envisagez de créer dans votre compte de gestion ou choisissez le nom d'un environnement existant qui nécessite une connexion à un compte d'environnement.

Créez une connexion à un compte d'environnement dans un compte d'environnement.

Exécutez la commande suivante :

```
$ 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"
```

Réponse :

```
{  
  "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"  
  }  
}
```

Acceptez ou refusez la connexion d'un compte d'environnement dans un compte de gestion, comme indiqué dans la commande et la réponse suivantes.

Note

Si vous refusez la connexion à ce compte d'environnement, vous ne serez pas en mesure d'accepter ou d'utiliser la connexion refusée au compte d'environnement.

Si vous spécifiez `Rejeter`, le statut passe de `en attente` pour `rejetés`.

Si vous spécifiez `Accepter`, le statut passe de `en attente` pour `connecté`.

Exécutez la commande suivante pour accepter la connexion au compte d'environnement :

```
$ aws proton accept-environment-account-connection \  
  --id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

Réponse :

```
{  
  "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"  
  }  
}
```

Exécutez la commande suivante pour rejeter la connexion au compte d'environnement :

```
$ aws proton reject-environment-account-connection \  
  --id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

Réponse :

```
{  
  "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"
  }
}

```

Afficher les connexions d'un compte d'environnement. Tu peux obtenir ou lister des connexions aux comptes d'environnement.

Exécutez la commande get suivante :

```

$ aws proton get-environment-account-connection \
  --id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"

```

Réponse :

```

{
  "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"
  }
}

```

Supprimez une connexion à un compte d'environnement dans un compte d'environnement.

Note

Si vous supprimez la connexion à ce compte d'environnement, AWS Proton ne sera pas en mesure de gérer les ressources de l'infrastructure d'environnement dans le

compte d'environnement tant qu'une nouvelle connexion à l'environnement n'aura pas été acceptée pour le compte d'environnement et l'environnement nommé. Vous êtes responsable du nettoyage des ressources provisionnées qui ne sont toujours pas connectées à l'environnement.

Exécutez la commande suivante :

```
$ aws proton delete-environment-account-connection \  
  --id "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

Réponse :

```
{  
  "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"  
  }  
}
```

Environnements gérés par le client

Avec les environnements gérés par le client, vous pouvez utiliser l'infrastructure existante, telle qu'un VPC, que vous avez déjà déployée en tant qu'AWS Proton environnement. Lorsque vous utilisez des environnements gérés par le client, vous pouvez provisionner vos propres ressources partagées en dehors de AWS Proton. Toutefois, vous pouvez toujours autoriser AWS Proton pour utiliser les sorties de provisionnement pertinentes en tant qu'entrées pour AWS Proton services lorsqu'ils sont déployés. Si les sorties peuvent changer, AWS Proton est en mesure d'accepter les mises à jour. AWS Proton n'est cependant pas en mesure de modifier directement l'environnement, étant donné que le provisionnement est géré en dehors de AWS Proton.

Une fois l'environnement créé, vous êtes responsable de fournir les mêmes résultats à AWS Proton qui aurait été créé si AWS Proton avait créé l'environnement, par exemple les noms de clusters Amazon ECS ou les identifiants Amazon VPC.

Grâce à cette fonctionnalité, vous pouvez déployer et mettre à jour AWS Proton ressources de service provenant d'un AWS Proton modèle de service pour cet environnement. Toutefois, l'environnement lui-même n'est pas modifié par les mises à jour des modèles dans AWS Proton. Vous êtes responsable de l'exécution des mises à jour de l'environnement et de la mise à jour de ces sorties dans AWS Proton.

Vous pouvez disposer de plusieurs environnements dans un seul compte, qui sont à la fois AWS Proton environnements gérés et gérés par le client. Vous pouvez également associer un deuxième compte et utiliser un AWS Proton modèle dans le compte principal pour exécuter les déploiements et les mises à jour des environnements et des services de ce second compte lié.

Comment utiliser les environnements gérés par le client

La première chose que les administrateurs doivent faire est d'enregistrer un modèle d'environnement importé et géré par le client. Ne fournissez pas de manifestes ni de fichiers d'infrastructure dans le bundle de modèles. Fournissez uniquement le schéma.

Le schéma ci-dessous présente une liste de sorties utilisant le format API ouvert et reproduit les sorties d'un AWS CloudFormation modèle.

Important

Seules les entrées de chaîne sont autorisées pour les sorties.

L'exemple suivant est un extrait des sections de sortie d'un AWS CloudFormation modèle pour un modèle Fargate correspondant.

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

Le schéma du fichier correspondant AWS Proton l'environnement importé est similaire au suivant. Ne fournissez pas de valeurs par défaut dans le schéma.

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

Au moment de l'enregistrement du modèle, vous indiquez que ce modèle est importé et que vous indiquez l'emplacement du compartiment Amazon S3 pour le bundle. AWS Proton vérifie que le schéma contient uniquement `environment_input_type` et non AWS CloudFormation paramètres du modèle avant de le mettre en brouillon.

Vous devez fournir les informations suivantes pour créer un environnement importé.

- Un rôle IAM à utiliser lors de déploiements.
- Une spécification avec les valeurs pour les sorties requises.

Vous pouvez fournir ces deux options via la console ou AWS CLI en utilisant un processus similaire au déploiement d'un environnement normal.

CodeBuild création d'un rôle de provisionnement

Des outils d'infrastructure en tant que code (iAAC) tels que AWS CloudFormation et Terraform nécessitent des autorisations pour les nombreux types de AWS ressources. Par exemple, si un modèle iAAC déclare un compartiment Amazon S3, il a besoin d'autorisations pour créer, lire, mettre à jour et supprimer des compartiments Amazon S3. Limiter les rôles aux autorisations minimales requises est considéré comme une bonne pratique en matière de sécurité. Compte tenu de l'ampleur de AWS ressources, il est difficile de créer des politiques de moindre privilège pour les modèles iAAC, en particulier lorsque les ressources gérées par ces modèles peuvent changer ultérieurement. Par exemple, lors des dernières modifications apportées à un modèle géré par AWS Proton, vous ajoutez une ressource de base de données RDS.

La configuration des autorisations appropriées permet de faciliter les déploiements de votre IaC. AWS Proton CodeBuild le provisionnement exécute des commandes CLI arbitraires fournies par le client dans un CodeBuild projet situé dans le compte du client. Généralement, ces commandes créent et suppriment une infrastructure à l'aide d'un outil d'infrastructure en tant que code (iAAC) tel que AWS CDK. Quand un AWS la ressource déploie dont le modèle utilise CodeBuild Approvisionnement, AWS va démarrer une compilation dans un CodeBuild projet géré par AWS. Un rôle est transmis à CodeBuild, qui CodeBuild suppose d'exécuter des commandes. Ce rôle, appelé CodeBuild Le rôle de provisionnement est fourni par le client et contient les autorisations requises pour provisionner l'infrastructure. Il est censé être assumé uniquement par CodeBuild et même AWS Proton Je ne peux pas le supposer.

Création du rôle

Le CodeBuild Le rôle de provisionnement peut être créé dans la console IAM ou dans AWS CLI. Pour le créer dans 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
```

Cela joint également le `AWSProtonCodeBuildProvisioningBasicAccess`, qui contient les autorisations minimales requises par CodeBuild service pour exécuter une compilation.

Si vous préférez utiliser la console, assurez-vous de ce qui suit lorsque vous créez le rôle :

1. Pour une entité de confiance, sélectionnez AWS service, puis sélectionnez CodeBuild.

2. À l'étape Ajouter des autorisations, sélectionnez `AWSProtonCodeBuildProvisioningBasicAccess` et toute autre politique que vous souhaitez joindre.

Accès administrateur

Si vous joignez la `AdministratorAccess` politique à l'égard du `CodeBuild` rôle de provisionnement, il garantira qu'aucun modèle iAAC n'échouera en raison d'un manque d'autorisations. Cela signifie également que toute personne capable de créer un modèle d'environnement ou un modèle de service peut effectuer des actions au niveau de l'administrateur, même si cet utilisateur n'est pas administrateur. AWS Proton ne recommande pas d'utiliser `AdministratorAccess` avec le `CodeBuild` rôle de provisionnement. Si vous décidez d'utiliser `AdministratorAccess` avec le `CodeBuild` rôle de provisionnement, faites-le dans un environnement sandbox.

Vous pouvez créer un rôle avec `AdministratorAccess` dans la console IAM ou en exécutant cette commande :

```
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
```

Création d'un rôle à portée minimale

Si vous souhaitez créer un rôle avec des autorisations minimales, il existe plusieurs approches :

- Déployez avec des autorisations d'administrateur, puis limitez le rôle. Nous vous recommandons d'utiliser [Analyseur d'accès IAM](#).
- Utilisez des politiques gérées pour donner accès aux services que vous prévoyez d'utiliser.

AWS CDK

Si vous utilisez AWS CDK avec AWS Proton, et tu as couru `cdk bootstrap` sur chaque compte/région d'environnement, alors il existe déjà un rôle pour `cdk deploy`. Dans ce cas, joignez la politique suivante au `CodeBuild` rôle de provisionnement :

```
{
  "Action": "sts:AssumeRole",
```

```

"Resource": [
  "arn:aws:iam::account-id:role/cdk-*-deploy-role-*",
  "arn:aws:iam::account-id:role/cdk-*-file-publishing-role-*"
],
"Effect": "Allow"
}

```

VPC personnalisé

Si tu décides de courirCodeBuilddans un[VPC personnalisé](#), vous aurez besoin des autorisations suivantes dans votreCodeBuildrôle :

```

{
  "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"
    }
  }
}
```

Vous pouvez également utiliser [AmazonEC2FullAccess](#) politique gérée, bien que cela inclue des autorisations dont vous n'avez peut-être pas besoin. Pour associer la politique gérée à l'aide de l'interface de ligne de commande :

```
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 services

Un AWS Proton service est une instantiation d'un modèle de service, comprenant normalement plusieurs instances de service et un pipeline. Une instance AWS Proton de service est une instantiation d'un modèle de service dans un [environnement](#) spécifique. Un modèle de service est une définition complète de l'infrastructure et du pipeline de services optionnels pour un AWS Proton service.

Une fois que vous avez déployé vos instances de service, vous pouvez les mettre à jour par le biais de poussées de code source qui déclenchent le pipeline CI/CD ou en mettant à jour le service vers de nouvelles versions de son modèle de service. AWS Proton vous avertit lorsque de nouvelles versions de son modèle de service seront disponibles afin que vous puissiez mettre à jour vos services vers une nouvelle version. Lorsque votre service est mis à jour, AWS Proton redéploie le service et les instances de service.

Ce chapitre explique comment gérer les services à l'aide des opérations de création, d'affichage, de mise à jour et de suppression. Pour plus d'informations, consultez le guide [The AWS Proton Service API Reference](#).

Rubriques

- [Créer un service](#)
- [Affichage des données de service](#)
- [Modifier un service](#)
- [Supprimer un service](#)
- [Afficher les données des instances de service](#)
- [Mise à jour d'une instance de service](#)
- [Mise à jour d'un pipeline de service](#)

Créer un service

Pour déployer une application avec AWS Proton, en tant que développeur, vous créez un service et fournissez les informations suivantes.

1. Le nom d'un modèle de AWS Proton service publié par l'équipe de la plateforme.
2. Un nom pour le service.

3. Nombre d'instances de service que vous souhaitez déployer.
4. Une sélection d'environnements que vous souhaitez utiliser.
5. Une connexion à votre référentiel de code si vous utilisez un modèle de service qui inclut un pipeline de services (facultatif).

Présentation d'un service ?

Lorsque vous créez un AWS Proton service, vous pouvez choisir parmi deux types de service différents :

- Modèle de service qui inclut un pipeline de service (par défaut).
- Un modèle de service qui n'inclut pas de pipeline de services.

Vous devez créer au moins une instance de service lors de la création de votre service.

Une instance de service et un pipeline optionnel sont associés à un service. Vous ne pouvez créer ou supprimer un pipeline que dans le contexte des actions de création et de suppression de services. Pour savoir comment ajouter et supprimer des instances d'un service, veuillez consulter [Modifier un service](#).

Note

Votre environnement est configuré pour un provisionnement autogéré ou pour un provisionnement autogéré. AWS Proton fournit des services dans un environnement en utilisant la même méthode de provisionnement que celle utilisée par l'environnement. Le développeur qui crée ou met à jour des instances de service ne voit pas la différence et son expérience est la même dans les deux cas.

Pour plus d'informations sur les méthodes de responsable, veuillez consulter [the section called "Méthodes de provisionnement"](#).

Modèles de service

Des versions majeures et mineures des modèles de service sont disponibles. Lorsque vous utilisez la console, vous sélectionnez la dernière version Recommended majeure et mineure du modèle de service. Lorsque vous utilisez le AWS CLI et que vous ne spécifiez que la version principale du modèle de service, vous indiquez implicitement sa dernière version Recommended mineure.

Ce qui suit décrit la différence entre les versions principales et secondaires des modèles et leur utilisation.

- Les nouvelles versions d'un modèle sont disponibles dès qu'elles sont approuvées par un membre de l'équipe de la plateforme. Cela signifie que de nouveaux services sont créés à l'aide de cette version et que vous êtes invité à mettre à jour les services existants vers la nouvelle version.
- L'AWS Proton équipe de la plateforme peut ainsi mettre automatiquement à jour les instances de service vers une nouvelle version mineure d'un modèle de service. Les versions mineures doivent être rétrocompatibles.
- Étant donné que les versions majeures nécessitent que vous fournissiez de nouvelles entrées dans le cadre du processus de mise à jour, vous devez mettre à jour votre service vers une version majeure de son modèle de service. Les versions majeures ne sont pas rétrocompatibles.

Créer un service

Les procédures suivantes montrent comment utiliser la AWS Proton console ou AWS CLI créer un service avec ou sans pipeline de services.

AWS Management Console

Créez un service comme indiqué dans les étapes suivantes de la console.

1. Dans la [AWS Proton console](#), choisissez Services.
2. Choisissez Créer un service.
3. Sur la page Choisir un modèle de service, sélectionnez un modèle et choisissez Configurer.

Si vous ne souhaitez pas utiliser un pipeline activé, choisissez un modèle marqué avec Exclut pipeline pour votre service.

4. Sur la page Configurer le service, dans la section Paramètres du service, entrez un nom de service.
5. (Facultatif) Saisissez une description du service.
6. Dans la section Paramètres du référentiel de services :
 - a. Pour CodeStar Connexion, choisissez votre connexion dans la liste.
 - b. Pour ID de référentiel, choisissez le nom de votre référentiel de code source dans la liste.

- c. Pour Nom de la branche, choisissez le nom de la branche de votre référentiel de code source dans la liste.
7. (Facultatif) Dans la section Balises, choisissez Ajouter une nouvelle balise et entrez une clé et une valeur pour créer une balise gérée par le client.
8. Choisissez Next (Suivant).
9. Sur la page Configurer les paramètres personnalisés, dans la section Instances de service, dans la section Nouvelle instance. Vous devez saisir des valeurs pour les `required` paramètres. Vous pouvez saisir des valeurs pour les `optional` paramètres ou utiliser les valeurs par défaut lorsqu'elles sont fournies.
10. Dans la section Entrées du pipeline, vous devez saisir des valeurs pour les `required` paramètres. Vous pouvez saisir des valeurs pour les `optional` paramètres ou utiliser les valeurs par défaut lorsqu'elles sont fournies.
11. Choisissez Suivant et passez en revue vos entrées.
12. Choisissez Create (Créer).

Consultez les détails et l'état du service, ainsi que les balises AWS gérées par les clients et les balises gérées par les clients pour votre service.

13. Dans le volet de navigation, choisissez Services.

Une nouvelle page affiche la liste de vos services ainsi que leur statut et d'autres détails.

AWS CLI

Lorsque vous utilisez le AWS CLI, vous spécifiez les entrées de service dans un `spec` fichier au format `YAML`. `aws-proton/service.yaml`, situé dans votre répertoire de code source.

Vous pouvez utiliser la `get-service-template-minor-version` commande CLI pour afficher le schéma requis et les paramètres facultatifs pour lesquels vous fournissez des valeurs dans votre fichier de spécifications.

Si vous souhaitez utiliser un modèle de service contenant `pipelineProvisioning: "CUSTOMER_MANAGED"`, n'incluez pas la `pipeline: section` dans votre spécification et n'incluez `-repository-connection-arn` pas de `-branch-name` paramètres dans votre `create-service` commande. `-repository-id`

Crée un service avec un pipeline de service comme illustré dans les étapes de la CLI suivantes.

1. Configurez le [rôle de service](#) pour le pipeline comme indiqué dans l'exemple de commande CLI suivant.

Commande :

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

2. La liste suivante présente un exemple de spécification, basé sur le schéma du modèle de service, qui inclut le pipeline de service et les entrées de l'instance.

Spécification :

```
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"
```

Créez un service avec un pipeline, comme indiqué dans l'exemple de commande et de réponse de la CLI suivant.

Commande :

```
$ 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"
```

Réponse :

```
{
  "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"
  }
}
```

Créez un service sans pipeline de service, comme indiqué dans l'exemple de commande et de réponse de la CLI suivant.

L'exemple suivant montre un exemple de spécification qui n'inclut pas les entrées du pipeline de service.

Spécification :

```
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"
```

Pour créer un service sans pipeline de service provisionné, vous devez indiquer le chemin d'accès à **aspec.yaml** et vous n'incluez pas les paramètres du référentiel, comme indiqué dans l'exemple de commande et de réponse de la CLI suivant.

Commande :

```
$ aws proton create-service \
```

```
--name "MySimpleServiceNoPipeline" \  
--template-major-version "1" \  
--template-name "fargate-service" \  
--spec "file://spec-no-pipeline.yaml"
```

Réponse :

```
{  
  "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"  
  }  
}
```

Affichage des données de service

Vous pouvez consulter et répertorier les données détaillées des services à l'aide de laAWS Proton console ou duAWS CLI.

AWS Management Console

Répertoriez et affichez les détails des services à l'aide de la [AWS Protonconsole](#), comme indiqué dans les étapes suivantes.

1. Pour consulter la liste de vos services, choisissez Services dans le volet de navigation.
2. Pour afficher les données détaillées, choisissez le nom d'un service.

Afficher les détails de votre service.

AWS CLI

Affichez les détails d'un service doté d'un pipeline de services, comme indiqué dans l'exemple de commande et de réponse de la CLI suivant.

Commande :

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

Réponse :

```
{
  "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"
  }
}
```

Affichez les détails d'un service sans pipeline de service, comme indiqué dans l'exemple de commande et de réponse de la CLI suivant.

Commande :

```
$ aws proton get-service \  
  --name "simple-svc-no-pipeline"
```

Réponse :

```
{  
  "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\nenvironment: 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"  
  }  
}
```

Modifier un service

Vous pouvez apporter les modifications suivantes à un AWS Proton service.

- Modifier la description du service.
- Modifiez un service en ajoutant et en supprimant des instances de service.

Modifier la description du service

Vous pouvez utiliser la console ou la AWS CLI pour modifier la description d'un service.

AWS Management Console

Modifiez un service à l'aide de la console comme décrit dans les étapes suivantes.

Dans la liste des services.

1. Dans la [AWS Protonconsole](#), choisissez Services.
2. Dans la liste des services, choisissez le bouton d'option situé à gauche du service à mettre à jour.
3. Choisissez Edit (Modifier).
4. Sur la page Configurer le service, remplissez le formulaire et choisissez Suivant.
5. Sur la page Configurer les paramètres personnalisés, choisissez Suivant.
6. Vérifiez vos modifications et choisissez Enregistrer les modifications.

Sur la page détaillée du service.

1. Dans la [AWS Protonconsole](#), choisissez Services.
2. Dans la liste des services, choisissez le nom du service à modifier.
3. Sur la page détaillée du service, choisissez Modifier.
4. Sur la page Configurer le service, remplissez le formulaire et choisissez Suivant.
5. Sur la page Configurer les paramètres personnalisés, remplissez le formulaire et choisissez Suivant.
6. Vérifiez vos modifications et choisissez Enregistrer les modifications.

AWS CLI

Modifiez une description comme indiqué dans l'exemple de commande et de réponse CLI suivant.

Commande :

```
$ aws proton update-service \  
  --name "MySimpleService" \  
  --description "Edit by updating description"
```

Réponse :

```
{  
  "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"
}
}
```

Modifier un service pour ajouter ou supprimer des instances de service

Pour un AWS Proton service, vous pouvez ajouter ou supprimer des instances de service en soumettant une spécification modifiée. Les conditions suivantes doivent être remplies pour que la demande soit acceptée :

- Votre service et votre pipeline ne sont pas déjà modifiés ou supprimés lorsque vous soumettez la demande de modification.
- Votre spécification modifiée n'inclut pas les modifications qui modifient le pipeline de services ni les modifications apportées aux instances de service existantes qui ne doivent pas être supprimées.
- Votre spécification modifiée ne supprime aucune instance de service existante associée à un composant. Pour supprimer une telle instance de service, vous devez d'abord mettre à jour le composant pour le détacher de son instance de service. Pour plus d'informations sur les composants, veuillez consulter [Composants](#).

Les instances dont la suppression a échoué sont des instances de service dans l'`DELETE_FAILED` état. Lorsque vous demandez une modification de service, AWS Proton tente de supprimer pour vous les instances dont la suppression a échoué, dans le cadre du processus de modification. Si l'une de vos instances de service ne parvient pas à être supprimée, il se peut que certaines ressources soient toujours associées à ces instances, même si elles ne sont pas visibles depuis la console ou AWS CLI. Vérifiez les ressources de votre infrastructure d'instance dont la suppression a échoué et nettoyez-les afin de pouvoir les supprimer pour vous.

Pour le quota d'instances de service pour un service, consultez [Quotas AWS Proton](#). Vous devez également gérer au moins une instance de service pour votre service après sa création. Au cours du processus de mise à jour, AWS Proton compte les instances de service existantes et les instances à

ajouter ou à supprimer. Les instances dont la suppression a échoué sont incluses dans ce décompte et vous devez en tenir compte lorsque vous modifiez vos `spec`.

Utiliser la console ou AWS CLI pour ajouter ou supprimer des instances de service

AWS Management Console

Modifiez votre service pour ajouter ou supprimer des instances de service à l'aide de la console.

Dans la [AWS Proton console](#)

1. Dans le volet de navigation, choisissez Services.
2. Sélectionnez le service que vous souhaitez modifier.
3. Choisissez Edit (Modifier).
4. (Facultatif) Sur la page Configurer le service, modifiez le nom ou la description du service, puis choisissez Suivant.
5. Sur la page Configurer les paramètres personnalisés, choisissez Supprimer pour supprimer une instance de service et choisissez Ajouter une nouvelle instance pour ajouter une instance de service et remplir le formulaire.
6. Choisissez Next (Suivant).
7. Vérifiez votre mise à jour et choisissez Enregistrer les modifications.
8. Un modal vous demande de vérifier la suppression des instances de service. Suivez les instructions et choisissez Oui, supprimer.
9. Sur la page détaillée du service, consultez les détails de l'état de votre service.

AWS CLI

Ajoutez et supprimez des instances de service avec une modification, `spec` comme indiqué dans les AWS CLI exemples de commandes et de réponses suivants.

Lorsque vous utilisez l'interface de ligne de commande, vos `spec` devez exclure les instances de service à supprimer et inclure à la fois les instances de service à ajouter et les instances de service existantes que vous n'avez pas marquées pour suppression.

La liste suivante montre l'exemple `spec` avant la modification et une liste des instances de service déployées par la spécification. Cette spécification a été utilisée dans l'exemple précédent pour modifier une description de service.

Spécification :

```

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"

```

L'exemple `delist-service-instances` commande et de réponse CLI suivant montre les instances actives avant d'ajouter ou de supprimer une instance de service.

Commande :

```

$ aws proton list-service-instances \
  --service-name "MySimpleService"

```

Réponse :

```

{
  "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"
    }
]
}

```

La liste suivante montre l'exemple modifié spec utilisé pour supprimer et ajouter une instance. L'instance nommée existant `my-instance` est supprimée et une nouvelle instance nommée `yet-another-instance` est ajoutée.

Spécification :

```

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"

```

Vous pouvez utiliser "\${Proton::CURRENT_VAL}" pour indiquer les valeurs de paramètres à conserver par rapport à l'originalspec, si les valeurs existent dans lespec. get-serviceUtilisez-lespec pour afficher l'original d'un service, comme décrit dans [Affichage des données de service](#).

La liste suivante montre comment vous pouvez vous "\${Proton::CURRENT_VAL}" assurer despec ne pas inclure les modifications des valeurs de paramètres pour que les instances de services existantes soient conservées.

Spécification :

```
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"
```

La liste suivante montre la commande CLI et la réponse pour modifier le service.

Commande :

```
$ aws proton update-service
  --name "MySimpleService" \
  --description "Edit by adding and deleting a service instance" \
  --spec "file://spec.yaml"
```

Réponse :

```
{
  "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"
  }
}

```

La `list-service-instances` commande et la réponse suivantes confirment que l'instance nommée `existing-instance` est supprimée et qu'une nouvelle instance nommée `yet-another-instance` est ajoutée.

Commande :

```

$ aws proton list-service-instances \
  --service-name "MySimpleService"

```

Réponse :

```

{
  "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"
  }
]
}
```

Ce qui se passe lorsque vous ajoutez ou supprimez des instances de service

Après avoir soumis une modification de service pour supprimer et ajouter des instances de service, AWS Proton prend les mesures suivantes.

- Définit le service sur `UPDATE_IN_PROGRESS`.
- Si le service possède un pipeline, définit son statut sur `IN_PROGRESS` et bloque les actions du pipeline.
- Définit toutes les instances de service à supprimer sur `DELETE_IN_PROGRESS`.
- Bloque les actions de service.
- Bloque les actions sur les instances de service qui sont marquées pour être supprimées.
- Crée de nouvelles instances de service.
- Supprime les instances que vous avez répertoriées pour suppression.
- Tentatives de suppression d'instances ayant échoué.
- Une fois les ajouts et les suppressions terminés, reprovise le pipeline de services (s'il y en a un), définit votre service `ACTIVE` et active les actions de service et de pipeline.

AWS Proton tente de remédier aux modes de défaillance comme suit.

- Si une ou plusieurs instances de service n'ont pas pu être créées, AWS Proton essaie de déprovisionner toutes les instances de service nouvellement créées et revient spéc à l'état précédent. Il ne supprime aucune instance de service et ne modifie en aucune façon le pipeline.

- Si une ou plusieurs instances de service n'ont pas pu être supprimées, AWS Proton reprovise le pipeline sans les instances supprimées. Le pipeline est mis à jour pour inclure les instances ajoutées et exclure celles qui ont été marquées pour suppression.
- Si le provisionnement du pipeline échoue, aucune annulation n'est tentée et le service et le pipeline reflètent un état de mise à jour échouée.

Étiquetage et modifications de service

Lorsque vous ajoutez des instances de service dans le cadre de la modification de votre service, les balises AWS gérées se propagent et sont automatiquement créées pour les nouvelles instances et les ressources provisionnées. Si vous créez de nouvelles balises, celles-ci ne sont appliquées qu'aux nouvelles instances. Les balises de service existantes gérées par le client se propagent également vers les nouvelles instances. Pour plus d'informations, consultez [AWS Proton ressources et balises](#).

Supprimer un service

Vous pouvez supprimer un AWS Proton service, avec ses instances et son pipeline, à l'aide de la AWS Proton console ou du AWS CLI.

Vous ne pouvez pas supprimer un service qui possède une instance de service associée à un composant. Pour supprimer un tel service, vous devez d'abord mettre à jour tous les composants attachés afin de les détacher de leurs instances de service. Pour plus d'informations sur les composants, veuillez consulter [Composants](#).

AWS Management Console

Supprimez un service à l'aide de la console comme décrit dans les étapes suivantes.

Sur la page détaillée du service.

1. Dans la [AWS Proton console](#), choisissez Services.
2. Dans la liste des services, choisissez le nom du service à supprimer.
3. Sur la page détaillée du service, choisissez Actions, puis Supprimer.
4. Un modal vous invite à confirmer l'action de suppression.
5. Suivez les instructions et choisissez Oui, supprimer.

AWS CLI

Supprimez un service comme indiqué dans l'exemple de commande et de réponse de la CLI suivant.

Commande :

```
$ aws proton delete-service \  
  --name "simple-svc"
```

Réponse :

```
{  
  "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"  
  }  
}
```

Afficher les données des instances de service

Apprenez à consulter les données détaillées des instances deAWS Proton service. Vous pouvez utiliser la console ou leAWS CLI.

Une instance de service appartient à un service. Vous ne pouvez créer ou supprimer une instance que dans le contexte des actions de [modification](#), de [création](#) et de [suppression](#) du service. Pour savoir comment ajouter et supprimer des instances d'un service, veuillez consulter [Modifier un service](#).

AWS Management Console

Répertoriez et affichez les détails des instances de service à l'aide de la [AWS Protonconsole](#), comme indiqué dans les étapes suivantes.

1. Pour consulter la liste de vos instances de service, choisissez Instances de services dans le volet de navigation.
2. Pour afficher les données détaillées, choisissez le nom d'une instance de service.

Consultez les données détaillées de votre instance de service.

AWS CLI

Répertoriez et affichez les détails des instances de service comme indiqué dans les exemples de commandes et de réponses de la CLI suivants.

Commande :

```
$ aws proton list-service-instances
```

Réponse :

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


Commande :

```
$ aws proton get-service-instance \
  --name "instance-one" \
  --service-name "simple-svc"
```

Réponse :

```
{
  "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"
  }
}
```

Mise à jour d'une instance de service

Apprenez à mettre à jour une instance AWS Proton de service et à annuler la mise à jour.

Une instance de service appartient à un service. Vous ne pouvez créer ou supprimer une instance que dans le contexte des actions de [modification](#), de [création](#) et de [suppression](#) du service. Pour savoir comment ajouter et supprimer des instances d'un service, veuillez consulter [Modifier un service](#).

Il existe quatre modes de mise à jour d'une instance de service, comme décrit dans la liste suivante. Lorsque vous utilisez le AWS CLI, le `deployment-type` champ définit le mode. Lorsque vous utilisez

la console, ces modes correspondent aux actions Modifier et Mettre à jour vers la dernière version mineure et Mettre à jour aux dernières actions de la version majeure qui figurent dans le menu Actions de la page détaillée de l'instance de service.

NONE

Dans ce mode, aucun déploiement n'a lieu. Seuls les paramètres de métadonnées demandés sont mis à jour.

CURRENT_VERSION

Dans ce mode, l'instance de service est déployée et mise à jour avec la nouvelle spécification que vous fournissez. Seuls les paramètres demandés sont mis à jour. N'incluez pas de paramètres de version mineurs ou majeurs lorsque vous l'utilisez `deployment-type`.

MINOR_VERSION

Dans ce mode, l'instance de service est déployée et mise à jour avec la version mineure publiée et recommandée (la plus récente) de la version majeure actuelle utilisée par défaut. Vous pouvez également spécifier une version mineure différente de la version majeure actuellement utilisée.

MAJOR_VERSION

Dans ce mode, l'instance de service est déployée et mise à jour avec la version majeure et mineure publiée et recommandée (la plus récente) du modèle actuel par défaut. Vous pouvez également spécifier une version majeure différente, supérieure à la version principale utilisée, ainsi qu'une version mineure (facultative).

Vous pouvez tenter d'annuler le déploiement d'une mise à jour d'une instance de service si `deploymentStatus` est le cas `IN_PROGRESS`. AWS Proton tente d'annuler le déploiement. L'annulation réussie n'est pas garantie.

Lorsque vous annulez le déploiement d'une mise à jour, AWS Proton tente d'annuler le déploiement comme indiqué dans les étapes suivantes.

- Définit l'état du déploiement sur `CANCELLING`.
- Arrête le déploiement en cours et supprime toutes les nouvelles ressources créées par le déploiement lorsque `IN_PROGRESS`.

- Définit l'état du déploiement sur CANCELLED.
- Rétablit l'état de la ressource à ce qu'il était avant le début du déploiement.

Pour plus d'informations sur l'annulation du déploiement d'une instance de service, consultez [CancelServiceInstanceDeployment](#) la référence AWS Proton API.

Utilisez la console ou AWS CLI pour effectuer des mises à jour ou annuler des déploiements de mises à jour.

AWS Management Console

Mettez à jour une instance de service à l'aide de la console en suivant ces étapes.

1. Dans la [AWS Proton console](#), choisissez Service instances dans le volet de navigation.
2. Dans la liste des instances de service, choisissez le nom de l'instance de service à mettre à jour.
3. Choisissez Actions, puis choisissez l'une des options de mise à jour, Modifier pour mettre à jour les spécifications ou les actions, puis Mettre à jour vers la dernière version mineure, ou Mettre à jour vers la dernière version majeure.
4. Remplissez chaque formulaire et cliquez sur Suivant jusqu'à ce que vous atteigniez la page de révision.
5. Vérifiez vos modifications et choisissez Mettre à jour.

AWS CLI

Mettez à jour une instance de service vers une nouvelle version mineure, comme indiqué dans les exemples de commandes et de réponses de la CLI.

Lorsque vous mettez à jour votre instance de service avec une modification spec, vous pouvez utiliser "`${Proton::CURRENT_VAL}`" pour indiquer les valeurs de paramètres à conserver par rapport à l'original spec, si les valeurs existent dans le spec. `get-service` Utilisez `lespec` pour afficher l'original d'une instance de service, comme décrit dans [Affichage des données de service](#).

L'exemple suivant montre comment utiliser "`${Proton::CURRENT_VAL}`" dans un spec.

Spécification :

```
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"

```

Commande : pour mettre à jour

```

$ 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"

```

Réponse :

```

{
  "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"
  }
}

```

```
}
}
```

Commande : pour obtenir et confirmer le statut

```
$ aws proton get-service-instance \
  --name "instance-one" \
  --service-name "simple-svc"
```

Réponse :

```
{
  "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 Management Console

Annulez le déploiement d'une instance de service à l'aide de la console, comme indiqué dans les étapes suivantes.

1. Dans la [AWS Proton console](#), choisissez Service instances dans le volet de navigation.

2. Dans la liste des instances de service, choisissez le nom de l'instance de service avec la mise à jour de déploiement que vous souhaitez annuler.
3. Si l'état du déploiement de vos mises à jour est En cours, sur la page détaillée de l'instance de service, choisissez Actions, puis Annuler le déploiement.
4. Un mode vous demande de confirmer l'annulation. Choisissez Annuler le déploiement.
5. L'état du déploiement de vos mises à jour est défini sur Annulation puis Annulé pour terminer l'annulation.

AWS CLI

Annulez la mise à jour du déploiement d'une instance de service IN_PROGRESS vers la nouvelle version mineure 2, comme indiqué dans les exemples de commandes et de réponses de la CLI suivants.

Une condition d'attente est incluse dans le modèle utilisé pour cet exemple afin que l'annulation commence avant la réussite du déploiement des mises à jour.

Commande : pour annuler

```
$ aws proton cancel-service-instance-deployment \  
  --service-instance-name "instance-one" \  
  --service-name "simple-svc"
```

Réponse :

```
{  
  "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:\nmy_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\nenvironment: MySimpleEnv\n spec:\n   my_sample_service_instance_required_input:\n   '789'\n",\n    "templateMajorVersion": "1",\n    "templateMinorVersion": "1",\n    "templateName": "svc-simple"\n  }\n}
```

Commande : pour obtenir et confirmer le statut

```
$ aws proton get-service-instance \
  --name "instance-one" \
  --service-name "simple-svc"
```

Réponse :

```
{\n  "serviceInstance": {\n    "arn": "arn:aws:proton:region-id:123456789012:service/simple-svc/service-\ninstance/instance-one",\n    "createdAt": "2021-04-02T21:29:59.962000+00:00",\n    "deploymentStatus": "CANCELLED",\n    "deploymentStatusMessage": "User initiated cancellation.",\n    "environmentName": "simple-env",\n    "lastDeploymentAttemptedAt": "2021-04-02T21:45:15.406000+00:00",\n    "lastDeploymentSucceededAt": "2021-04-02T21:38:00.823000+00:00",\n    "name": "instance-one",\n    "serviceName": "simple-svc",\n    "spec": "proton: ServiceSpec\n\npipeline:\nmy_sample_pipeline_optional_input: \"abc\"\n my_sample_pipeline_required_input:\n\"123\"\n\ninstances:\n - name: \"instance-one\"\n   environment: \"simple-\nenv\"\n   spec:\n     my_sample_service_instance_optional_input: \"def\"\n     my_sample_service_instance_required_input: \"456\"\n - name: \"my-\nother-instance\"\n   environment: \"kls-simple-env\"\n   spec:\n     my_sample_service_instance_required_input: \"789\"",\n    "templateMajorVersion": "1",\n    "templateMinorVersion": "1",\n    "templateName": "svc-simple"\n  }\n}
```

Mise à jour d'un pipeline de service

Apprenez à mettre à jour un pipeline AWS Proton de services et à annuler la mise à jour.

Un pipeline de service appartient à un service. Vous ne pouvez créer ou supprimer un pipeline que dans le contexte des actions de [création](#) et de [suppression](#) de services.

Il existe quatre modes de mise à jour d'un pipeline de service, comme décrit dans la liste suivante. Lorsque vous utilisez le AWS CLI, le `deployment-type` champ définit le mode. Lorsque vous utilisez la console, ces modes correspondent au pipeline d'édition et Update à la version recommandée.

NONE

Dans ce mode, aucun déploiement n'a lieu. Seuls les paramètres de métadonnées demandés sont mis à jour.

CURRENT_VERSION

Dans ce mode, le pipeline de services est déployé et mis à jour avec la nouvelle spécification que vous fournissez. Seuls les paramètres demandés sont mis à jour. N'incluez pas de paramètres de version mineurs ou majeurs lorsque vous l'utilisez `deployment-type`.

MINOR_VERSION

Dans ce mode, le pipeline de services est déployé et mis à jour avec la version mineure publiée et recommandée (la plus récente) de la version majeure actuelle utilisée par défaut. Vous pouvez également spécifier une version mineure différente de la version majeure actuellement utilisée.

MAJOR_VERSION

Dans ce mode, le pipeline de services est déployé et mis à jour avec la version majeure et mineure publiée et recommandée (la plus récente) du modèle actuel par défaut. Vous pouvez également spécifier une version majeure différente, supérieure à la version principale utilisée, ainsi qu'une version mineure (facultative).

Vous pouvez tenter d'annuler le déploiement d'une mise à jour d'un pipeline de services si `deploymentStatus` est le cas `IN_PROGRESS`. AWS Proton tente d'annuler le déploiement. L'annulation réussie n'est pas garantie.

Lorsque vous annulez le déploiement d'une mise à jour, AWS Proton tente d'annuler le déploiement comme indiqué dans les étapes suivantes.

- Définit l'état du déploiement sur CANCELLING.
- Arrête le déploiement en cours et supprime toutes les nouvelles ressources créées par le déploiement lorsque IN_PROGRESS.
- Définit l'état du déploiement sur CANCELLED.
- Rétablit l'état de la ressource à ce qu'il était avant le début du déploiement.

Pour plus d'informations sur l'annulation du déploiement d'un pipeline de services, consultez [CancelServicePipelineDeployment](#) la référence AWS Proton API.

Utilisez la console ou AWS CLI pour effectuer des mises à jour ou annuler des déploiements de mises à jour.

AWS Management Console

Mettez à jour un pipeline de service à l'aide de la console, comme décrit dans les étapes suivantes.

1. Dans la [AWS Proton console](#), choisissez Services.
2. Dans la liste des services, choisissez le nom du service pour lequel vous souhaitez mettre à jour le nom du pipeline.
3. La page détaillée du service comporte deux onglets : Aperçu et Pipeline. Choisissez Pipeline.
4. Si vous souhaitez mettre à jour les spécifications, choisissez Modifier le pipeline et remplissez chaque formulaire, puis choisissez Suivant jusqu'à ce que vous ayez rempli le formulaire final, puis choisissez Mettre à jour le pipeline.

Si vous souhaitez effectuer une mise à jour vers une nouvelle version et qu'une icône d'information indique qu'une nouvelle version est disponible dans le modèle Pipeline, choisissez le nom de la nouvelle version du modèle.

- a. Choisissez Mettre à jour vers la version recommandée.
- b. Remplissez chaque formulaire et cliquez sur Suivant jusqu'à ce que vous remplissiez le formulaire final et que vous choisissiez Mettre à jour.

AWS CLI

Mettez à jour un pipeline de service vers une nouvelle version mineure, comme indiqué dans les exemples de commandes et de réponses de la CLI suivants.

Lorsque vous mettez à jour votre pipeline de service avec une `modificationspec`, vous pouvez utiliser `"${Proton::CURRENT_VAL}"` pour indiquer les valeurs de paramètres à conserver par rapport à l'`originalspec`, si les valeurs existent dans `lespec`. Utilisez `get-service` pour afficher l'original d'un pipeline de service, comme décrit dans [Affichage des données de service](#).

L'exemple suivant montre comment utiliser `"${Proton::CURRENT_VAL}"` dans `unspec`.

Spécification :

```
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"
```

Commande : pour mettre à jour

```
$ 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"
```

Réponse :

```
{
  "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\n",
    "templateMajorVersion": "1",
    "templateMinorVersion": "0",
    "templateName": "svc-simple"
  }
}
```

Commande : pour obtenir et confirmer le statut

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

Réponse :

```
{
  "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 Management Console

Annulez le déploiement d'un pipeline de services à l'aide de la console, comme indiqué dans les étapes suivantes.

1. Dans la [AWS Protonconsole](#), choisissez Services dans le volet de navigation.
2. Dans la liste des services, sélectionnez le nom du service dont la mise à jour de déploiement que vous souhaitez annuler.
3. Sur la page des détails du service, choisissez l'onglet Pipeline.
4. Si l'état du déploiement de vos mises à jour est En cours, sur la page détaillée du pipeline de services, choisissez Annuler le déploiement.
5. Un mode vous demande de confirmer l'annulation. Choisissez Annuler le déploiement.
6. L'état du déploiement de vos mises à jour est défini sur Annulation puis Annulé pour terminer l'annulation.

AWS CLI

Annulez une mise à jour du déploiement du pipeline de service IN_PROGRESS vers la version mineure 2, comme indiqué dans les exemples de commandes et de réponses de la CLI suivants.

Une condition d'attente est incluse dans le modèle utilisé pour cet exemple afin que l'annulation commence avant la réussite du déploiement des mises à jour.

Commande : pour annuler

```
$ aws proton cancel-service-pipeline-deployment \  
  --service-name "simple-svc"
```

Réponse :

```
{  
  "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"  
  }  
}
```

Commande : pour obtenir et confirmer le statut

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

Réponse :

```
{  
  "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"
  }
}

```

Composants AWS Proton

Les composants sont un type deAWS Proton ressource. Ils ajoutent de la flexibilité aux modèles de service. Les composants fournissent aux équipes de plateforme un mécanisme leur permettant d'étendre les modèles d'infrastructure de base et de définir des mesures de protection qui permettent aux développeurs de gérer certains aspects de leur infrastructure d'applications.

DansAWS Proton les administrateurs, les administrateurs définissent l'infrastructure standard utilisée par les équipes de développement et les applications. Toutefois, les équipes de développement peuvent avoir besoin d'inclure des ressources supplémentaires pour leurs cas d'utilisation spécifiques, comme les files d'attente Amazon Simple Queue Service (Amazon SQS) ou les tables Amazon DynamoDB. Ces ressources spécifiques aux applications peuvent changer fréquemment, en particulier au début du développement d'applications. Le maintien de ces modifications fréquentes dans les modèles créés par les administrateurs peut s'avérer difficile à gérer et à faire évoluer. Les administrateurs auraient besoin de gérer de nombreux modèles supplémentaires sans véritable valeur ajoutée pour les administrateurs. L'alternative, qui consiste à laisser les développeurs d'applications créer des modèles pour leurs applications, n'est pas idéale non plus, car elle empêche les administrateurs de standardiser les principaux composants de l'architecture, tels queAWS Fargate les tâches. C'est là que les composants entrent en jeu.

Avec un composant, un développeur peut ajouter des ressources supplémentaires à son application, au-delà de ce que les administrateurs ont défini dans les modèles d'environnement et de service. Le développeur attache ensuite le composant à une instance de service. AWS Protonfournit les ressources d'infrastructure définies par le composant tout comme il fournit des ressources pour les environnements et les instances de service.

Un composant peut lire les entrées d'une instance de service et fournir des sorties à l'instance de service, pour une expérience totalement intégrée. Par exemple, si le composant ajoute un compartiment Amazon Simple Storage Service (Amazon S3) destiné à être utilisé par une instance de service, le modèle de composant peut prendre en compte les noms de l'environnement et des instances de service pour nommer le compartiment. Lors duAWS Proton rendu du modèle de service pour provisionner une instance de service, l'instance de service peut faire référence au bucket et l'utiliser.

Les composantsAWS Proton actuellement pris en charge sont des composants directement définis. Vous transmettez le fichier d'infrastructure sous forme de code (IaC) qui définit l'infrastructure du composant directement à l'AWS ProtonAPI ou à la console. Cela est différent d'un environnement ou d'un service, dans lequel vous définissez l'IaC dans un ensemble de modèles et enregistrez le bundle

en tant que ressource modèle, puis utilisez une ressource modèle pour créer l'environnement ou le service.

Note

Les composants directement définis permettent aux développeurs de définir une infrastructure supplémentaire et de la mettre en service. AWS Proton provisionne tous les composants directement définis s'exécutant dans le même environnement en utilisant le même rôle AWS Identity and Access Management (IAM).

Un administrateur peut contrôler ce que les développeurs peuvent faire avec les composants de deux manières :

- Sources de composants prises en charge : un administrateur peut autoriser l'attachement de composants à des instances de service en fonction d'une propriété des versions des modèles de AWS Proton service. Par défaut, les développeurs ne peuvent pas associer de composants à des instances de service.


Pour plus d'informations sur cette propriété, consultez le [supportedComponentSources](#) paramètre de l'action [CreateServiceTemplateVersion](#) API dans la référence de l'AWS Proton API.

Note

Lorsque vous utilisez la synchronisation des modèles, AWS Proton crée implicitement des versions de modèles de service lorsque vous apportez des modifications à un ensemble de modèles de services dans un référentiel. Dans ce cas, au lieu de spécifier les sources de composants prises en charge lors de la création de la version du modèle de service, vous spécifiez cette propriété dans un fichier associé à chaque version principale du modèle de service. Pour plus d'informations, consultez [the section called "Modèles de service de synchronisation"](#).

- Rôles des composants : un administrateur peut attribuer un rôle de composant à un environnement. AWS Proton assume ce rôle lorsqu'il fournit une infrastructure définie par un composant directement défini dans l'environnement. Par conséquent, le rôle du composant limite l'infrastructure que les développeurs peuvent ajouter à l'aide de composants directement définis dans l'environnement. En l'absence du rôle des composants, les développeurs ne peuvent pas créer de composants directement définis dans l'environnement.

Pour plus d'informations sur l'attribution d'un rôle à un composant, consultez le [componentRoleArn](#) paramètre de l'action d'[CreateEnvironment](#) API dans la référence AWS Proton API.

 Note

Les rôles des composants ne sont pas utilisés dans [Approvisionnement](#) les environnements.

Rubriques

- [Comment les composants se comparent-ils aux autres AWS Proton ressources ?](#)
- [Composants de la AWS Proton console](#)
- [Les composants de l'AWS Proton API et AWS CLI](#)
- [Questions fréquentes sur les composants](#)
- [États des composants](#)
- [Infrastructure de composants sous forme de fichiers de code](#)
- [AWS CloudFormation Exemple de composants](#)

Comment les composants se comparent-ils aux autres AWS Proton ressources ?

À bien des égards, les composants sont similaires aux autres AWS Proton ressources. Leur infrastructure est définie dans un [fichier modèle IAC](#), créé au format AWS CloudFormation YAML ou Terraform HCL. AWS Proton peut provisionner l'infrastructure des composants à l'aide du [provisionnement AWS géré](#) ou du [provisionnement autogéré](#).

Les composants se distinguent toutefois des autres AWS Proton ressources de plusieurs manières :

- **État détaché** : les composants sont conçus pour être attachés à des instances de service et pour étendre leur infrastructure, mais ils peuvent également être dans un état détaché, c'est-à-dire qu'ils ne sont attachés à aucune instance de service. Pour de plus amples informations sur les états des composants, veuillez consulter [the section called “États des composants”](#).

- Pas de schéma : aucun schéma n'est associé aux composants, contrairement aux [ensembles de modèles](#). Les entrées des composants sont définies par un service. Un composant peut consommer des entrées lorsqu'il est attaché à une instance de service.
- Aucun composant géré par le client : AWS Proton provisionne toujours l'infrastructure des composants pour vous. Il n'existe pas de version des composants utilisant vos propres ressources. Pour de plus amples informations sur les environnements gérés par le client, veuillez consulter [the section called "Création"](#).
- Aucune ressource modèle : les composants directement définis ne sont pas associés à une ressource modèle similaire aux modèles d'environnement et de service. Vous fournissez un fichier modèle IAC directement au composant. De même, vous fournissez directement un manifeste qui définit le langage du modèle et le moteur de rendu pour le provisionnement de l'infrastructure du composant. Vous créez le fichier modèle et le manifeste de la même manière que vous créez un [ensemble de modèles](#). Toutefois, avec des composants directement définis, il n'est pas nécessaire de stocker les fichiers iAC sous forme de lots à des emplacements particuliers, et vous ne créez pas de modèle de ressource à AWS Proton partir de fichiers iAC.
- Provisionnement non CodeBuild basé : vous ne pouvez pas provisionner des composants directement définis à l'aide de votre propre script de provisionnement personnalisé, connu sous le nom de provisionnement CodeBuild basé. Pour plus d'informations, consultez [the section called "CodeBuild provisionnement"](#).

Composants de la AWS Proton console

Utilisez la AWS Proton console pour créer, mettre à jour, afficher et utiliser AWS Proton des composants.

Les pages de console suivantes concernent les composants. Nous incluons des liens directs vers les pages de console de niveau supérieur.

- [Composants](#) : consultez la liste des composants de votre AWS compte. Vous pouvez créer de nouveaux composants et mettre à jour ou supprimer des composants existants. Choisissez le nom d'un composant dans la liste pour afficher la page contenant ses détails.

Des listes similaires existent également sur les pages Détails de l'environnement et Détails de l'instance de service. Ces listes affichent uniquement les composants associés à la ressource consultée. Lorsque vous créez un composant à partir de l'une de ces listes, AWS Proton présélectionne l'environnement associé sur la page Créer un composant.

- **Détails du composant** — Pour afficher la page de détails du composant, choisissez un nom de composant dans la liste des [composants](#).

Sur la page de détails, consultez les détails et l'état du composant, puis mettez à jour ou supprimez le composant. Affichez et gérez les listes de sorties (par exemple, les ARN de ressources provisionnées), les AWS CloudFormation piles provisionnées et les balises attribuées.

- **Créer un composant** : crée un composant. Entrez le nom et la description du composant, choisissez les ressources associées, spécifiez le fichier IAC source du composant et attribuez des balises.
- **Mettre à jour le composant** : pour mettre à jour un composant, sélectionnez-le dans la liste des [composants](#), puis, dans le menu Actions, choisissez Mettre à jour le composant. Vous pouvez également choisir Mettre à jour sur les pages de détails du composant.

Vous pouvez mettre à jour la plupart des détails du composant. Vous ne pouvez pas mettre à jour le nom du composant. Et vous pouvez choisir de redéployer ou non le composant après une mise à jour réussie.

- **Configurer l'environnement** : lorsque vous créez ou mettez à jour un environnement, vous pouvez spécifier un rôle de composant. Ce rôle contrôle la capacité à exécuter des composants directement définis dans l'environnement et fournit des autorisations pour leur provisionnement.
- **Créer une nouvelle version de modèle de service** : lorsque vous créez une version de modèle de service, vous pouvez spécifier les sources de composants prises en charge pour la version du modèle. Cela contrôle la possibilité d'attacher des composants à des instances de service de services en fonction de cette version de modèle.

Les composants de l'AWS ProtonAPI etAWS CLI

Utilisez l'AWS ProtonAPI ou laAWS CLI pour créer, mettre à jour, afficher et utiliserAWS Proton des composants.

Les actions d'API suivantes gèrent directement les ressources desAWS Proton composants.

- [CreateComponent](#)— Créez unAWS Proton composant.
- [DeleteComponent](#)— Supprime unAWS Proton composant.
- [GetComponent](#)— Obtenez des données détaillées pour un composant.
- [ListComponentOutputs](#)— Obtenez une liste des sorties de l'infrastructure sous forme de code (IaC) des composants.

- [ListComponentProvisionedResources](#)— Répertorie les ressources provisionnées pour un composant avec des détails.
- [ListComponents](#)— Répertorie les composants avec des données récapitulatives. Vous pouvez filtrer la liste de résultats par environnement, service ou instance de service unique.

Les actions d'API suivantes d'autres AWS Proton ressources possèdent certaines fonctionnalités liées aux composants.

- [CreateEnvironment](#), [UpdateEnvironment](#)—`componentRoleArn` À utiliser pour spécifier le nom de ressource Amazon (ARN) du rôle de service IAM AWS Proton utilisé lors du provisionnement de composants directement définis dans cet environnement. Il détermine l'étendue de l'infrastructure qu'un composant directement défini peut fournir.
- [CreateServiceTemplateVersion](#)—`supportedComponentSources` À utiliser pour spécifier les sources de composants prises en charge. Les composants dont les sources sont prises en charge peuvent être attachés à des instances de service en fonction de cette version de modèle de service.

Questions fréquentes sur les composants

Quel est le cycle de vie d'un composant ?

Les composants peuvent être attachés ou détachés. Ils sont conçus pour être attachés à une instance de service et améliorer son infrastructure la plupart du temps. Les composants détachés sont dans un état de transition qui vous permet de supprimer un composant ou de l'attacher à une autre instance de service de manière contrôlée et sécurisée. Pour plus d'informations, consultez [the section called “États des composants”](#).

Pourquoi ne puis-je pas supprimer mes composants attachés ?

Solution : pour supprimer un composant attaché, mettez à jour le composant pour le détacher de l'instance de service, validez la stabilité de l'instance de service, puis supprimez le composant.

Pourquoi est-ce nécessaire ? Les composants connectés fournissent l'infrastructure supplémentaire dont votre application a besoin pour exécuter ses fonctions d'exécution. L'instance de service peut utiliser les sorties des composants pour détecter et utiliser les ressources de cette infrastructure. La suppression du composant, donc de ses ressources d'infrastructure, peut perturber l'instance de service attachée.

Par mesure de sécurité supplémentaire, AWS Proton vous devez mettre à jour le composant et le détacher de son instance de service avant de pouvoir le supprimer. Vous pouvez ensuite valider votre instance de service pour vous assurer qu'elle continue à se déployer et à fonctionner correctement. Si vous détectez un problème, vous pouvez rapidement rattacher le composant à l'instance de service, puis vous efforcer de le résoudre. Lorsque vous êtes certain que votre instance de service est exempte de toute dépendance vis-à-vis du composant, vous pouvez supprimer le composant en toute sécurité.

Pourquoi ne puis-je pas modifier directement l'instance de service attachée à un composant ?

Solution : pour modifier la pièce jointe, mettez à jour le composant pour le détacher de l'instance de service, validez la stabilité du composant et de l'instance de service, puis attachez le composant à la nouvelle instance de service.

Pourquoi est-ce nécessaire ? Un composant est conçu pour être attaché à une instance de service. Votre composant peut utiliser des entrées d'instance de service pour la dénomination et la configuration des ressources d'infrastructure. La modification de l'instance de service attachée peut perturber le composant (en plus d'une éventuelle interruption de l'instance de service), comme décrit dans la FAQ précédente, [Pourquoi ne puis-je pas supprimer mes composants attachés ?](#)). Par exemple, cela peut entraîner le changement de nom, voire le remplacement, des ressources définies dans le modèle IaC du composant.

Par mesure de sécurité supplémentaire, AWS Proton vous devez mettre à jour le composant et le détacher de son instance de service avant de pouvoir l'attacher à une autre instance de service. Vous pouvez ensuite valider la stabilité du composant et de l'instance de service avant d'attacher le composant à la nouvelle instance de service.

États des composants

AWS Proton les composants peuvent se trouver dans deux états fondamentalement différents :

- **Attaché** : le composant est attaché à une instance de service. Il définit l'infrastructure qui prend en charge les fonctionnalités d'exécution de l'instance de service. Le composant étend l'infrastructure définie dans les modèles d'environnement et de service avec une infrastructure définie par le développeur.

Un composant typique est attaché pendant la majeure partie de son cycle de vie utile.

- **Détaché** : le composant est associé à un AWS Proton environnement et n'est rattaché à aucune instance de service de l'environnement.

Il s'agit d'un état transitoire qui permet de prolonger la durée de vie d'un composant au-delà d'une seule instance de service.

Le tableau suivant fournit une comparaison de haut niveau des différents états des composants.

	Attached	Detached
Objectif principal de l'État	Pour étendre l'infrastructure d'une instance de service.	Maintenir l'infrastructure du composant entre les attachements des instances de service.
Associé à	Une instance de service et un environnement	Un environnement
Propriétés spécifiques clés	<ul style="list-style-type: none"> • Nom du service • Nom de l'instance de service • Spec 	<ul style="list-style-type: none"> • Nom de l'environnement
Peut être supprimé	× Non	✓ Oui
Peut être mis à jour vers une autre instance de service	× Non	✓ Oui
Peut lire les entrées	✓ Oui	× Non

L'objectif principal d'un composant est d'être rattaché à une instance de service et d'étendre son infrastructure avec des ressources supplémentaires. Un composant attaché peut lire les entrées de l'instance de service conformément à la spécification. Vous ne pouvez pas supprimer directement le composant ni l'attacher à une autre instance de service. Vous ne pouvez pas non plus supprimer son instance de service ni le service et l'environnement associés. Pour effectuer l'une de ces opérations, mettez d'abord à jour le composant pour le détacher de son instance de service.

Pour maintenir l'infrastructure du composant au-delà de la durée de vie d'une instance de service unique, vous mettez à jour le composant et le détachez de son instance de service en supprimant le service et les noms des instances de service. Cet État détaché est un État de transition. Le composant n'a aucune entrée. Son infrastructure reste provisionnée et vous pouvez la mettre à jour. Vous pouvez supprimer les ressources auxquelles le composant était associé lors de son attachement (instance de service, service). Vous pouvez supprimer le composant ou le mettre à jour pour le rattacher à nouveau à une instance de service.

Infrastructure de composants sous forme de fichiers de code

L'infrastructure des composants sous forme de fichiers de code (IAC) est similaire à celle des autres AWS Proton ressources. Découvrez ici certains détails spécifiques aux composants. Pour des informations complètes sur la création de fichiers iAC pour AWS Proton, consultez [Création de modèles et d'ensembles](#).

Utilisation de paramètres avec des composants

L'espace de noms des AWS Proton paramètres inclut certains paramètres auxquels un fichier IAC de service peut faire référence pour obtenir le nom et les sorties d'un composant associé. L'espace de noms inclut également des paramètres auxquels un fichier IAC de composant peut faire référence pour obtenir les entrées, les sorties et les valeurs de ressources de l'environnement, du service et de l'instance de service auxquels le composant est associé.

Un composant ne possède pas ses propres entrées : il reçoit ses entrées de l'instance de service à laquelle il est attaché. Un composant peut également lire les sorties de l'environnement.

Pour plus d'informations sur l'utilisation de paramètres dans les fichiers IAC des composants et des services associés, reportez-vous à la section [the section called "Paramètres du composant CloudFormation IaC"](#). Pour des informations générales sur AWS Proton les paramètres et une référence complète de l'espace de noms des paramètres, reportez-vous à la section [the section called "Paramètres"](#).

Création de fichiers iAC robustes

En tant qu'administrateur, lorsque vous créez une version de modèle de service, vous pouvez décider si vous souhaitez autoriser les instances de service créées à partir de la version du modèle à avoir des composants attachés. Consultez le [supportedComponentSources](#) paramètre de l'action d'[CreateServiceTemplateVersion](#) API dans la référence de AWS Proton l'API. Toutefois, pour toute future instance de service, la personne qui crée l'instance, décide d'y attacher ou non un composant

et (dans le cas de composants définis directement) est généralement une personne différente : un développeur utilisant votre modèle de service. Par conséquent, vous ne pouvez pas garantir qu'un composant soit attaché à une instance de service. Vous ne pouvez pas non plus garantir l'existence de noms de sortie de composants spécifiques ou la validité et la sécurité des valeurs de ces sorties.

AWS Proton et la syntaxe Jinja vous aident à résoudre ces problèmes et à créer des modèles de service robustes qui s'affichent sans défaillance de la manière suivante :

- **AWS Proton filtres de paramètres** : lorsque vous faites référence aux propriétés de sortie des composants, vous pouvez utiliser des filtres de paramètres, c'est-à-dire des modificateurs qui valident, filtrent et mettent en forme les valeurs des paramètres. Pour plus d'informations et d'exemples, consultez [the section called "CloudFormation filtres de paramètres"](#).
- **Propriété unique par défaut** : lorsque vous faites référence à une ressource ou à une propriété de sortie unique d'un composant, vous pouvez garantir que le rendu de votre modèle de service n'échouera pas en utilisant le `default` filtre, avec ou sans valeur par défaut. Si le composant, ou un paramètre de sortie spécifique auquel vous faites référence, n'existe pas, la valeur par défaut (ou une chaîne vide, si vous n'avez pas spécifié de valeur par défaut) est affichée à la place, et le rendu est réussi. Pour plus d'informations, consultez [the section called "Fournir des valeurs par défaut"](#).

Exemples :

- `{{ service_instance.components.default.name | default("") }}`
- `{{ service_instance.components.default.outputs.my-output | default("17") }}`

Note

Ne confondez pas la `.default` partie de l'espace de noms, qui désigne les composants directement définis, avec le `default` filtre, qui fournit une valeur par défaut lorsque la propriété référencée n'existe pas.

- **Référence complète de l'objet** : lorsque vous faites référence à l'ensemble du composant, ou à la collection des sorties d'un composant, AWS Proton renvoie un objet vide et garantit ainsi que le rendu de votre modèle de service n'échouera pas. `{}` Vous ne devez pas utiliser de filtre. Veillez à faire la référence dans un contexte pouvant contenir un objet vide, ou utilisez une `{{ if .. }}` condition pour tester la présence d'un objet vide.

Exemples :

- `{{ service_instance.components.default }}`
- `{{ service_instance.components.default.outputs }}`

AWS CloudFormationExemple de composants

Voici un exemple complet d'un composantAWS Proton directement défini et de la manière dont vous pouvez l'utiliser dans unAWS Proton service. Le composant fournit un compartiment Amazon Simple Storage Service (Amazon S3) et la politique d'accès associée. L'instance de service peut faire référence à ce bucket et l'utiliser. Le nom du compartiment est basé sur les noms de l'environnement, du service, de l'instance de service et du composant, ce qui signifie que le compartiment est couplé à une instance spécifique du modèle de composant étendant une instance de service spécifique. Les développeurs peuvent créer plusieurs composants sur la base de ce modèle de composant, afin de fournir des compartiments Amazon S3 pour différentes instances de service et différents besoins fonctionnels.

L'exemple couvre la création des différentesAWS CloudFormation infrastructures requises sous forme de fichiers de code (IAC) et la création d'un rôle obligatoireAWS Identity and Access Management (IAM). L'exemple regroupe les étapes en fonction des rôles des personnes propriétaires.

Étapes de l'administrateur

Pour permettre aux développeurs d'utiliser des composants avec un service

1. Créez un rôleAWS Identity and Access Management (IAM) qui limite les ressources que les composants directement définis exécutés dans votre environnement peuvent fournir. AWS Protonassume ce rôle ultérieurement pour fournir des composants directement définis dans l'environnement.

Pour cet exemple, utilisez la stratégie suivante :

Exemple rôle de composant directement défini

```
{
  "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"
    }
  }
}
]
}

```

- Indiquez le rôle que vous avez créé à l'étape précédente lors de la création ou de la mise à jour de l'environnement. Dans la AWS Proton console, spécifiez un rôle de composant sur la page

Configurer l'environnement. Si vous utilisez l'AWS ProtonAPI ouAWS CLI si vous spécifiez les actionscomponentRoleArn de [UpdateEnvironment](#)l'API. [CreateEnvironment](#)

3. Créez un modèle de service qui fait référence à un composant directement défini attaché à l'instance de service.

L'exemple montre comment écrire un modèle de service robuste qui ne se casse pas si un composant n'est pas attaché à l'instance de service.

Exemple fichier CloudFormation IAC de service utilisant un composant

```
# 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. Créez une nouvelle version mineure du modèle de service qui déclare que les composants directement définis sont pris en charge.
 - Ensemble de modèles dans Amazon S3 : dans laAWS Proton console, lorsque vous créez une version de modèle de service, dans la zone Sources de composants prises en charge, choisissez Directement défini. Si vous utilisez l'AWS ProtonAPI ouAWS CLI, spécifiezDIRECTLY_DEFINED dans lesupportedComponentSources paramètre des actions [CreateServiceTemplateVersion](#)ou de [UpdateServiceTemplateVersion](#)l'API.
 - Synchronisation des modèles : apportez une modification à votre référentiel d'ensembles de modèles de services, que vous spécifiezsupported_component_sources : enDIRECTLY_DEFINED tant qu'élément du .template-registration.yaml fichier dans le répertoire des versions principales. Pour de plus amples informations sur ce fichier, veuillez consulter [the section called "Modèles de service de synchronisation"](#).
5. Publiez la version mineure du nouveau modèle de service. Pour plus d'informations, consultez [the section called "Publier"](#).
6. Veillez à autoriser lesproton:CreateComponent développeurs qui utilisent ce modèle de service à jouer le rôle IAM.

Étapes du développeur

Pour utiliser un composant directement défini avec une instance de service

1. Créez un service qui utilise la version du modèle de service que l'administrateur a créée avec la prise en charge des composants. Vous pouvez également mettre à jour l'une de vos instances de service existantes pour utiliser la dernière version du modèle.
2. Rédigez un fichier modèle IAC composant qui provisionne un compartiment Amazon S3 et une politique d'accès associée et expose ces ressources en tant que sorties.

Exemple fichier CloudFormation IAC du composant

```
# 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. Si vous utilisez l'AWS ProtonAPI ou AWS CLI si vous écrivez un fichier manifeste pour le composant.

Exemple manifeste de composant directement défini


```

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

```

4. Créez un composant directement défini. AWS Proton assume le rôle de composant que l'administrateur a défini pour approvisionner le composant.

Dans la AWS Proton console, sur la page [Composants](#), choisissez Créer un composant. Pour les paramètres du composant, entrez un nom de composant et une description du composant (facultatif). Pour Attacher un composant, choisissez Attacher le composant à une instance de service. Sélectionnez votre environnement, votre service et votre instance de service. Pour Source du composant, choisissez AWS CloudFormation, puis choisissez le fichier IAC du composant.

 Note

Vous n'avez pas besoin de fournir un manifeste : la console en crée un pour vous.

Si vous utilisez l'AWS Proton API ou AWS CLI si vous utilisez l'action [CreateComponent](#) API. Définissez un composant `name` et optionnel `description`.
Set `environmentName` `serviceName`, et `serviceInstanceName`.
Définissez `templateSource` et `manifest` accédez aux chemins des fichiers que vous avez créés.

 Note

La spécification d'un nom d'environnement est facultative lorsque vous spécifiez des noms de service et d'instance de service. La combinaison de ces deux éléments est unique dans votre AWS compte et AWS Proton peut déterminer l'environnement à partir de l'instance de service.

5. Mettez à jour votre instance de service pour la redéployer. AWS Proton utilise les sorties de votre composant dans le modèle d'instance de service rendu, afin de permettre à votre application d'utiliser le compartiment Amazon S3 que le composant a provisionné.

Utiliser des référentiels Git avec AWS Proton

AWS Proton utilise les référentiels Git à des fins diverses. La liste suivante classe les types de référentiels associés aux AWS Proton ressources. Pour les AWS Proton fonctionnalités qui se connectent à plusieurs reprises à votre référentiel pour y envoyer du contenu ou en extraire du contenu, vous devez enregistrer un lien vers le référentiel AWS Proton dans votre AWS compte. Un lien de référentiel est un ensemble de propriétés qui AWS Proton peuvent être utilisées lors de la connexion à un référentiel. AWS Proton prend actuellement GitHub en charge GitHub Enterprise et BitBucket.

Référentiels pour développeurs

Référentiel de code : référentiel que les développeurs utilisent pour stocker le code de l'application. Utilisé pour le déploiement du code. AWS Proton n'interagit pas directement avec ce référentiel. Lorsqu'un développeur met en service un service qui inclut un pipeline, il fournit le nom du référentiel et la branche à partir desquels lire le code de son application. AWS Proton transmet ces informations au pipeline qu'il approvisionne.

Pour plus d'informations, veuillez consulter [the section called "Création"](#).

Référentiels d'administrateur

Référentiel de modèles : référentiel dans lequel les administrateurs stockent des ensembles de AWS Proton modèles. Utilisé pour la synchronisation des modèles. Lorsqu'un administrateur crée un modèle dans AWS Proton, il peut pointer vers un référentiel de modèles AWS Proton et synchroniser le nouveau modèle avec celui-ci. Lorsque l'administrateur met à jour le bundle de modèles dans le référentiel, il crée AWS Proton automatiquement une nouvelle version du modèle. Liez un référentiel de modèles à AWS Proton avant de pouvoir l'utiliser pour la synchronisation.

Pour plus d'informations, veuillez consulter [the section called "Configurations de synchronisation de modèles"](#).

Note

Un référentiel de modèles n'est pas nécessaire si vous continuez à télécharger vos modèles sur Amazon Simple Storage Service (Amazon S3) et à appeler les API de gestion des AWS Proton modèles pour créer de nouveaux modèles ou de nouvelles versions de modèles.

Répartiments autogérés

Référentiel d'infrastructure : référentiel qui héberge des modèles d'infrastructure rendus. Utilisé pour le provisionnement autogéré de l'infrastructure de ressources. Lorsqu'un administrateur crée un environnement pour un provisionnement autogéré, il fournit un référentiel. AWS Proton soumet des pull requests (PR) à ce référentiel afin de créer l'infrastructure pour l'environnement et pour toute instance de service déployée dans l'environnement. Liez un référentiel d'infrastructure AWS Proton avant de pouvoir l'utiliser pour le provisionnement d'une infrastructure autogérée.

Référentiel de pipelines : référentiel utilisé pour créer des pipelines. Utilisé pour le provisionnement autogéré des pipelines. L'utilisation d'un référentiel supplémentaire pour provisionner des pipelines permet AWS Proton de stocker les configurations de pipeline indépendamment de tout environnement ou service individuel. Il vous suffit de fournir un référentiel de pipeline unique pour tous vos services de provisionnement autogérés. Liez un référentiel de pipeline à AWS Proton avant de pouvoir l'utiliser pour le provisionnement autogéré du pipeline.

Pour plus d'informations, veuillez consulter [the section called “AWS-provisionnement géré”](#).

Rubriques

- [Créer un lien vers votre référentiel](#)
- [Afficher les données du référentiel lié](#)
- [Suppression d'un lien vers un référentiel](#)

Créer un lien vers votre référentiel

Vous pouvez créer un lien vers votre référentiel au moyen de l'interface I ou de la console ou de la console ou de l'interface I ou de l'interface Lorsque vous créez un lien vers un référentiel, un [rôle lié à un service est AWS Proton](#) créé pour vous.

AWS Management Console

Créer un lien vers votre référentiel, comme indiqué dans les étapes de console suivantes.

1. Dans la [AWS Proton console](#), choisissez Référentiels.
2. Choisissez Créer un référentiel.
3. Sur la page Lier un nouveau référentiel, dans la section Détails du référentiel :

- a. Choisissez votre fournisseur de référentiel.
 - b. Choisissez l'une de vos connexions existantes. Si vous n'en avez pas, choisissez Ajouter une nouvelleCodeStar connexion pour créer une connexion, puis revenez à laAWS Proton console, actualisez la liste des connexions et choisissez votre nouvelle connexion.
 - c. Choisissez parmi vos référentiels de code source connectés.
4. [facultatif] Dans la section Tags, choisissez Ajouter un nouveau tag une ou plusieurs fois et entrez des paires clé/valeur.
 5. Choisissez Créer un référentiel.
 6. Consultez les données détaillées de votre référentiel lié.

AWS CLI

Créez et enregistrez un lien vers votre référentiel.

Exécutez la commande suivante :

```
$ 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
```

Les deux derniers paramètres,--encryption-key et--tags, sont facultatifs.

Réponse :

```
{  
  "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"  
  }  
}
```

Après avoir créé un lien vers un référentiel, vous pouvez consulter la [AWS liste des balises gérées par le client](#), comme indiqué dans l'exemple de commande suivant. AWS Proton génère automatiquement des balises AWS gérées pour vous. Vous pouvez également modifier et créer des balises gérées par le client à l'aide du AWS CLI. Pour plus d'informations, veuillez consulter [AWS Proton ressources et balises](#).

Commande :

```
$ aws proton list-tags-for-resource \  
    --resource-arn "arn:aws:proton:region-id:123456789012:repository/github:myrepos/  
environments"
```

Afficher les données du référentiel lié

Vous pouvez répertorier et afficher les détails des référentiels liés à l'aide de la console ou du AWS CLI. Pour les liens de référentiel utilisés pour synchroniser les référentiels Git AWS Proton, vous pouvez récupérer la définition et l'état de synchronisation du référentiel à l'aide du AWS CLI.

AWS Management Console

Répertoriez et affichez les détails des référentiels liés à l'aide de la [AWS Proton console](#).

1. Pour répertorier vos référentiels liés, choisissez Référentiels dans le volet de navigation.
2. Pour afficher les données détaillées, choisissez le nom d'un référentiel.

AWS CLI

Répertoriez vos référentiels liés.

Exécutez la commande suivante :

```
$ aws proton list-repositories
```

Réponse :

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

Afficher les détails d'un référentiel lié.

Exécutez la commande suivante :

```
$ aws proton get-repository \
  --name myrepos/templates \
  --provider "GITHUB"
```

Réponse :

```
{
  "repository": {
    "arn": "arn:aws:proton:region-id:123456789012:repository/github:myrepos/
templates",
    "name": "myrepos/templates",
    "provider": "GITHUB"
  }
}
```

Répertoriez vos référentiels synchronisés.

L'exemple suivant répertorie les référentiels que vous avez configurés pour la synchronisation des modèles.

Exécutez la commande suivante :

```
$ aws proton list-repository-sync-definitions \  
  --branch "main" \  
  --repository-name myrepos/templates \  
  --repository-provider "GITHUB" \  
  --sync-type "TEMPLATE_SYNC"
```

Afficher l'état de synchronisation du référentiel.

L'exemple suivant permet de récupérer le statut de synchronisation d'un référentiel de synchronisation d'un modèle.

Exécutez la commande suivante :

```
$ aws proton get-repository-sync-status \  
  --branch "main" \  
  --repository-name myrepos/templates \  
  --repository-provider "GITHUB" \  
  --sync-type "TEMPLATE_SYNC"
```

Réponse :

```
{  
  "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"
  }
}
```

Suppression d'un lien vers un référentiel

Vous pouvez supprimer un lien vers un référentiel à l'aide de la console ou de laAWS CLI.

Note

La suppression d'un lien vers un référentiel supprime uniquement le lienAWS Proton enregistré qui se trouve dans votreAWS compte. Il ne supprime aucune information de votre référentiel.

AWS Management Console

Supprimez un lien vers un référentiel à l'aide de la console.

Sur la page détaillée du référentiel.

1. Dans la [AWS Protonconsole](#), choisissez Référentiels.
2. Dans la liste Compartiments, choisissez le bouton radio situé à gauche du nom du nom du nom du nom que vous souhaitez supprimer.
3. Choisissez Delete (Supprimer).
4. Un nom vous demande de confirmer l'action Supprimer.
5. Suivez les instructions et choisissez Oui, supprimer.

AWS CLI

Supprimez un lien vers un référentiel.

Exécutez la commande suivante :

```
$ aws proton delete-repository \
  --name myrepos/templates \
  --provider"GITHUB"
```

Réponse :

```
{
  "repository": {
    "arn": "arn:aws:proton:region-id:123456789012:repository/github:myrepos/
templates",
    "name": "myrepos/templates",
    "provider": "GITHUB"
  }
}
```

Surveillance AWS Proton

La surveillance joue un rôle important dans le maintien de la fiabilité, de la disponibilité AWS Proton et des performances de vos AWS solutions. La section suivante décrit les outils de surveillance que vous pouvez utiliser AWS Proton.

Automatisez AWS Proton avec EventBridge

Vous pouvez suivre AWS Proton les événements sur Amazon EventBridge. EventBridge fournit un flux de données en temps réel à partir de vos propres applications, applications software-as-a-service (SaaS) et Services AWS. Vous pouvez configurer des événements pour répondre aux modifications de l'état des AWS ressources. EventBridge achemine ensuite ces données vers des services cibles tels qu' AWS Lambda Amazon Simple Notification Service. Ces événements sont les mêmes que ceux qui apparaissent dans Amazon CloudWatch Events. CloudWatch Les événements fournissent un flux d'événements système en temps quasi réel qui décrivent les modifications apportées aux AWS ressources. Pour plus d'informations, consultez [Qu'est-ce qu'Amazon EventBridge ?](#) dans le guide de EventBridge l'utilisateur Amazon.

EventBridge À utiliser pour être informé des changements d'état dans les flux de travail de AWS Proton provisionnement.

Types d'événements

Les événements sont composés de règles qui incluent un modèle d'événements et des cibles. Vous configurez une règle en choisissant le modèle d'événement et les objets cibles :

Modèle d'événement

Chaque règle est exprimée sous la forme d'un modèle d'événements avec la source et le type d'événements à surveiller ainsi que les cibles des événements. Pour surveiller les événements, vous créez une règle avec le service que vous surveillez comme source d'événements. Par exemple, vous pouvez créer une règle avec un modèle d'événement utilisé AWS Proton comme source d'événement pour déclencher une règle en cas de modification de l'état d'un déploiement.

Cibles

La règle reçoit un service sélectionné comme cible de l'événement. Vous pouvez configurer un service cible pour envoyer des notifications, capturer des informations d'état, prendre des mesures correctives, initier des événements ou prendre d'autres mesures.

Les objets d'événement contiennent des champs standard tels que l'ID, le compte Région AWS, le type de détail, la source, la version, la ressource, l'heure (facultatif). Le champ de détail est un objet imbriqué contenant des champs personnalisés pour l'événement.

AWS Proton les événements sont émis dans la mesure du possible. La fourniture de tous les efforts signifie que le service tente d'envoyer tous les événements à EventBridge, mais dans de rares cas, un événement peut ne pas être livré.

Pour chaque AWS Proton ressource pouvant émettre des événements, le tableau suivant répertorie la valeur du type de détail, les champs de détail et (le cas échéant) une référence à une liste de valeurs pour les champs de `previousStatus` détail `status` et de détail. Lorsqu'une ressource est supprimée, la valeur du champ de `status` détail est `DELETED`.

Ressource	Valeur du type de détail	Champs de détail
<code>EnvironmentTemplate</code>	AWS Proton Modification du statut du modèle d'environnement	<code>name</code> <code>status</code> <code>previousStatus</code>
<code>EnvironmentTemplateVersion</code>	AWS Proton Modification de l'état de la version du modèle d'environnement	<code>name</code> <code>majorVersion</code> <code>minorVersion</code> <code>status</code> <code>previousStatus</code> valeurs de statut
<code>ServiceTemplate</code>	AWS Proton Modification du statut du modèle de service	<code>name</code> <code>status</code>

Ressource	Valeur du type de détail	Champs de détail
		previousStatus
ServiceTemplateVersion	AWS Proton Modification du statut de la version du modèle de service	name majorVersion minorVersion status previousStatus valeurs de statut
Environment	AWS Proton Modification de l'état de l'environnement	name status previousStatus
Service	AWS Proton Modification de l'état du service	name status previousStatus valeurs de statut

Ressource	Valeur du type de détail	Champs de détail
ServiceInstance	AWS Proton Modification du statut de l'instance de service	name serviceName status previousStatus
ServicePipeline	AWS Proton Modification de l'état du pipeline de services	serviceName status previousStatus
EnvironmentAccount Connection	AWS Proton Modification de l'état de connexion au compte d'environnement	id status previousStatus valeurs de statut
Component	AWS Proton Modification de l'état du composant	name status previousStatus

AWS Proton exemples d'événements

Les exemples suivants montrent AWS Proton comment envoyer des événements à EventBridge.

Modèle de service

```
{
  "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 du modèle de service

```
{
  "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"
  }
}
```

Environnement

```
{
  "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: Envoyer des alertes Amazon Simple Notification Service en cas de modification AWS Proton de l'état du service

Dans ce didacticiel, vous allez utiliser une règle d'événement AWS Proton préconfigurée qui capture les modifications de statut de votre AWS Proton service. EventBridgeenvoie les modifications de statut à une rubrique Amazon SNS. Vous vous abonnez au sujet et Amazon SNS vous envoie des e-mails de modification du statut de votre AWS Proton service.

Prérequis

Vous avez un AWS Proton service existant avec un `Active` statut. Dans le cadre de ce didacticiel, vous pouvez ajouter des instances de service à ce service, puis les supprimer.

Si vous devez créer un AWS Proton service, consultez [Démarrer](#). Pour plus d'informations, consultez [Quotas AWS Proton](#) et [the section called "Modifier"](#).

Étape 1 : Créer une rubrique Amazon SNS et s'y abonner

Créez une rubrique Amazon SNS qui servira de cible d'événement pour la règle d'événement que vous créez à l'étape 2.

Créer une rubrique Amazon SNS

1. Connectez-vous et ouvrez la console [Amazon SNS](#).
2. Dans le volet de navigation, choisissez Rubriques, puis Créer une rubrique.
3. Dans la page Créer un sujet :
 - a. Choisissez le type Standard.
 - b. Dans Nom, entrez **tutorial-service-status-change** et choisissez Créer un sujet.
4. Sur la page tutorial-service-status-changedétaillée, choisissez Créer un abonnement.
5. Sur la page Créer un abonnement :
 - a. Pour Protocole, choisissez E-mail.

- b. Dans le champ Endpoint (Point de terminaison), saisissez l'adresse e-mail à laquelle vous avez actuellement accès et choisissez Create subscription (Créer un abonnement).
6. Vérifiez votre compte de messagerie et attendez de recevoir un e-mail de confirmation de l'abonnement. Lorsque vous le recevez, ouvrez-le et choisissez Confirmer l'abonnement.

Étape 2 : Enregistrer une règle d'événement

Enregistrez une règle d'événement qui capture les modifications de statut de votre AWS Proton service. Pour de plus amples informations, veuillez consulter [Prérequis](#).

Créez une règle d'événement.

1. Ouvrez la [EventBridge console Amazon](#).
2. Dans le volet de navigation, choisissez Events, Rules.
3. Sur la page Règles, dans la section Règles, choisissez Créer une règle.
4. Sur la page Créer une règle :
 - a. Dans la section Nom et description, pour Nom, entrez **tutorial-rule**.
 - b. Dans la section Définir le modèle, choisissez le modèle d'événement.
 - i. Dans Modèle de correspondance d'événement, choisissez Prédéfini par un service.
 - ii. Pour Service Provider (Fournisseur de service), sélectionnez AWS.
 - iii. Pour Nom du service, choisissez AWS Proton.
 - iv. Pour Type d'événement, choisissez Changement AWS Proton de statut du service.

Le modèle d'événement apparaît dans un éditeur de texte.
5. Ouvrez la [AWS Proton console](#).
6. Dans le panneau de navigation, choisissez Services.
7. Dans la page Services, choisissez le nom de votre AWS Proton service.
8. Sur la page des détails du service, copiez le nom de ressource Amazon (ARN) du service.
9. Revenez à la EventBridge console et à la règle de votre didacticiel, puis choisissez Modifier dans l'éditeur de texte.
10. Dans l'éditeur de texte, pour "resources" :, entrez l'ARN du service que vous avez copié à l'étape viii.

```
{
  "source": ["aws.proton"],
  "detail-type": ["AWS Proton Service Status Change"],
  "resources": ["arn:aws:proton:region-id:123456789012:service/your-service"]
}
```

- xi. Enregistrez le modèle d'événement.
- c. Dans la section Sélectionner les cibles :
 - i. Pour Target (Cible), choisissez SNS topic (Rubrique SNS).
 - ii. Dans le champ Sujet, sélectionnez tutorial-service-status-change.
- d. Choisissez Créer.

Étape 3 : Testez la règle de votre événement

Vérifiez que votre règle d'événement fonctionne en ajoutant une instance à votre AWS Proton service.

1. Passez à la [AWS Proton console](#).
2. Dans le panneau de navigation, choisissez Services.
3. Dans la page Services, choisissez le nom de votre service.
4. Sur la page Détails du service, choisissez Modifier.
5. Dans la page Configurer le service, choisissez Next.
6. Dans la page Configurer les paramètres personnalisés, dans la section Instances de service, choisissez Ajouter une nouvelle instance.
7. Complétez le formulaire pour votre nouvelle instance :
 - a. Entrez un nom pour votre nouvelle instance.
 - b. Sélectionnez les mêmes environnements compatibles que ceux que vous avez choisis pour vos instances existantes.
 - c. Entrez des valeurs pour les entrées requises.
 - d. Choisissez Suivant.
8. Passez en revue vos entrées et choisissez Mettre à jour.

9. Une fois l'état du service activé `Active`, consultez vos e-mails pour vérifier que vous avez reçu AWS des notifications contenant des mises à jour du statut.

```
{
  "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"
  }
}
```

Étape 4 : Nettoyer

Supprimez votre rubrique Amazon SNS et votre abonnement, puis supprimez votre EventBridge règle.

Supprimez votre rubrique Amazon SNS et votre abonnement.

1. Accédez à la [console Amazon SNS](#).
2. Dans le volet de navigation, choisissez Abonnements.
3. Sur la page Abonnements, choisissez l'abonnement que vous avez souscrit à la rubrique nommée, `tutorial-service-status-change` puis choisissez Supprimer.
4. Dans le panneau de navigation, sélectionnez Sujets.
5. Sur la page Sujets, choisissez le sujet nommé, `tutorial-service-status-change` puis sélectionnez Supprimer.
6. Un modal vous invite à vérifier la suppression. Suivez les instructions et choisissez Supprimer.

Supprimez votre EventBridge règle.

1. Accédez à la [EventBridge console Amazon](#).
2. Dans le volet de navigation, choisissez Events, Rules.
3. Sur la page Règles, choisissez la règle nommée, `tutorial-rule` puis sélectionnez Supprimer.
4. Un modal vous invite à vérifier la suppression. Sélectionnez Delete (Supprimer).

Supprimez l'instance de service ajoutée.

1. Accédez à la [console AWS Proton](#).
2. Dans le panneau de navigation, choisissez Services.
3. Sur la page Services, choisissez le nom de votre service.
4. Sur la page des détails du service, choisissez Modifier, puis Suivant.
5. Sur la page Configurer les paramètres personnalisés, dans la section Instances de service, choisissez Supprimer pour l'instance de service que vous avez créée dans le cadre de ce didacticiel, puis cliquez sur Suivant.
6. Passez en revue vos entrées et choisissez Mettre à jour.
7. Un modal vous invite à vérifier la suppression. Suivez les instructions et choisissez Oui, supprimer.

Maintenez l'infrastructure à jour grâce au AWS Proton tableau de bord

Le AWS Proton tableau de bord fournit un résumé des AWS Proton ressources de votre AWS compte, en mettant particulièrement l'accent sur l'obsolescence, c'est-à-dire sur la manière dont les ressources déployées sont mises à jour. Une ressource déployée est à jour lorsqu'elle utilise la version recommandée du modèle associé. Une ressource out-of-date déployée peut nécessiter une mise à jour majeure ou mineure de la version du modèle.

Afficher le tableau de bord dans la AWS Proton console

Pour afficher le AWS Proton tableau de bord, ouvrez la [AWS Proton console](#), puis, dans le volet de navigation, sélectionnez Tableau de bord.

Ressources

The screenshot shows the AWS Proton Dashboard with the following data:

Resources			
Service instances	Services	Environments	Components
2	1	1	0

Resource templates	
Resource type	Total
Service templates	1
Environment templates	1

Resource status summary				
Resource type	Up to date	Failed	Minor update pending	Major update pending
Services	1	0	0	0
Service instances	2	0	0	0
Environments	1	0	0	0
Components	0	0	0	0

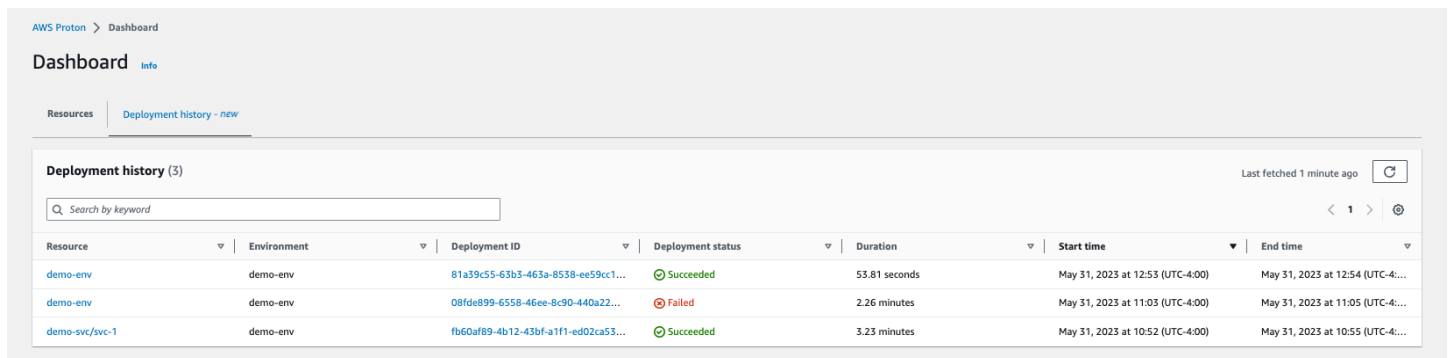
Service instances (11)						
Name	Deployment status	Service template	Service	Environment	Last successful deployment	Created
demo-inst-2	Succeeded	demo-svc-temp-lambda	demo-svc-lambda	demo-env-lambda	May 31, 2023 at 10:52 (UTC-4:00)	May 31, 2023 at 10:52 (UTC-4:00)
demo-inst-1	Succeeded	demo-svc-temp-lambda	demo-svc-lambda	demo-env-lambda	May 31, 2023 at 10:52 (UTC-4:00)	May 31, 2023 at 10:52 (UTC-4:00)

Le premier onglet du tableau de bord affiche le décompte de toutes les ressources de votre compte. L'onglet Ressources indique le nombre d'instances de service, de services, d'environnements et de composants, ainsi que vos modèles de ressources. Il décompose également le nombre de ressources pour chaque type de ressource déployé en fonction de l'état des ressources de ce type. Un tableau des instances de service présente les détails de chaque instance de service : son statut de déploiement, les AWS Proton ressources auxquelles elle est associée, les mises à jour disponibles et certains horodatages.

Vous pouvez filtrer la liste des instances de service en fonction de n'importe quelle propriété de table. Par exemple, vous pouvez filtrer pour voir les instances de service dont les déploiements se situent dans une fenêtre de temps spécifique, ou les instances de service obsolètes par rapport aux recommandations de versions majeures ou mineures.

Choisissez le nom d'une instance de service pour accéder à la page détaillée de l'instance de service, où vous pouvez agir pour effectuer les mises à jour de version appropriées. Choisissez un autre nom de AWS Proton ressource pour accéder à sa page détaillée, ou choisissez un type de ressource pour accéder à la liste de ressources correspondante.

Historique de déploiement



The screenshot shows the AWS Proton console interface. At the top, there is a breadcrumb 'AWS Proton > Dashboard' and a 'Dashboard' header with an 'Info' link. Below this, there are tabs for 'Resources' and 'Deployment history - new'. The main content area is titled 'Deployment history (3)' and includes a search bar with the placeholder 'Search by keyword'. To the right of the search bar, it says 'Last fetched 1 minute ago' and has a refresh icon. Below the search bar is a table with the following columns: Resource, Environment, Deployment ID, Deployment status, Duration, Start time, and End time. The table contains three rows of deployment data.

Resource	Environment	Deployment ID	Deployment status	Duration	Start time	End time
demo-env	demo-env	81a39c55-63b3-463a-8538-ee59cc1...	Succeeded	53.81 seconds	May 31, 2023 at 12:53 (UTC-4:00)	May 31, 2023 at 12:54 (UTC-4:00)
demo-env	demo-env	08fde899-6558-46ee-8c90-440a22...	Failed	2.26 minutes	May 31, 2023 at 11:03 (UTC-4:00)	May 31, 2023 at 11:05 (UTC-4:00)
demo-svc-1	demo-env	fb60af89-4b12-43bf-a1f1-ed02ca53...	Succeeded	3.23 minutes	May 31, 2023 at 10:52 (UTC-4:00)	May 31, 2023 at 10:55 (UTC-4:00)

L'onglet Historique des déploiements vous permet de consulter les détails de vos déploiements. Dans le tableau de l'historique des déploiements, vous pouvez suivre l'état du déploiement, ainsi que l'environnement et l'ID de déploiement. Vous pouvez choisir le nom de la ressource ou l'ID de déploiement pour obtenir encore plus de détails, tels qu'un message d'état du déploiement et les sorties de ressources. Le tableau vous permet également de filtrer selon n'importe quelle propriété du tableau.

Sécurité dans AWS Proton

Chez AWS, la sécurité dans le cloud est notre priorité numéro 1. En tant que client AWS, vous bénéficiez de centres de données et d'architectures réseau conçus pour répondre aux exigences des organisations les plus pointilleuses en termes de sécurité.

La sécurité est une responsabilité partagée entre AWS et vous. Le [modèle de responsabilité partagée](#) décrit cela comme la sécurité du cloud et la sécurité dans le cloud :

- Sécurité du cloud – AWS est responsable de la protection de l'infrastructure qui exécute des Services AWS dans le AWS Cloud. AWS vous fournit également les services que vous pouvez utiliser en toute sécurité. Des auditeurs tiers testent et vérifient régulièrement l'efficacité de notre sécurité dans le cadre des [AWS programmes de conformité](#). Pour en savoir plus sur les programmes de conformité qui s'appliquent à AWS Proton, reportez-vous aux [Services AWS concernés par le programme de conformité](#).
- Sécurité dans le cloud – Votre responsabilité est fonction du Service AWS que vous utilisez. Vous êtes également responsable d'autres facteurs, y compris de la sensibilité de vos données, des exigences de votre entreprise, ainsi que de la législation et de la réglementation applicables.

Cette documentation vous aide à comprendre comment appliquer le modèle de responsabilité partagée lors de l'utilisation de AWS Proton. Les rubriques suivantes expliquent comment configurer AWS Proton pour répondre à vos objectifs de sécurité et de conformité. Vous pouvez également apprendre à utiliser d'autres Services AWS capables de vous aider à surveiller et à sécuriser vos ressources AWS Proton.

Rubriques

- [Identity and Access Management \(Gestion des identités et des accès\) pour AWS Proton](#)
- [Configuration et analyse des vulnérabilités dans AWS Proton](#)
- [Protection des données dans AWS Proton](#)
- [Sécurité de l'infrastructure dans AWS Proton](#)
- [Journalisation et surveillance dans AWS Proton](#)
- [Résilience dans AWS Proton](#)
- [Bonnes pratiques de sécurité pour AWS Proton](#)
- [Prévention du député confus entre services](#)
- [CodeBuild mise en place d'un support Amazon VPC personnalisé](#)

Identity and Access Management (Gestion des identités et des accès) pour AWS Proton

AWS Identity and Access Management (IAM) est un Service AWS qui aide un administrateur à contrôler en toute sécurité l'accès aux ressources AWS. Des administrateurs IAM contrôlent les personnes qui s'authentifient (sont connectées) et sont autorisées (disposent d'autorisations) à utiliser des ressources AWS Proton. IAM est un Service AWS que vous pouvez utiliser sans frais supplémentaires.

Rubriques

- [Public ciblé](#)
- [Authentification par des identités](#)
- [Gestion des accès à l'aide de politiques](#)
- [Fonctionnement de AWS Proton avec IAM](#)
- [Exemples de politiques pour AWS Proton](#)
- [Politiques gérées par AWS pour AWS Proton](#)
- [Utilisation des rôles liés aux services pour AWS Proton](#)
- [Résolution des problèmes d'identité et d'accès avec AWS Proton](#)

Public ciblé

Votre utilisation d'AWS Identity and Access Management (IAM) diffère selon la tâche que vous accomplissez dans AWS Proton.

Utilisateur du service – Si vous utilisez le service AWS Proton pour effectuer votre tâche, votre administrateur vous fournit les informations d'identification et les autorisations dont vous avez besoin. Plus vous utiliserez de fonctions AWS Proton pour effectuer votre travail, plus vous pourriez avoir besoin d'autorisations supplémentaires. En comprenant bien la gestion des accès, vous saurez demander les autorisations appropriées à votre administrateur. Si vous ne pouvez pas accéder à une fonctionnalité dans AWS Proton, consultez [Résolution des problèmes d'identité et d'accès avec AWS Proton](#).

Administrateur du service – Si vous êtes le responsable des ressources AWS Proton de votre entreprise, vous bénéficiez probablement d'un accès total à AWS Proton. Votre responsabilité

est de déterminer AWS Proton les fonctionnalités ainsi que les ressources auxquelles les utilisateurs de votre service doivent accéder. Vous devez ensuite soumettre les demandes à votre administrateur IAM pour modifier les autorisations des utilisateurs de votre service. Consultez les informations sur cette page pour comprendre les concepts de base d'IAM. Pour en savoir plus sur la façon dont votre entreprise peut utiliser IAM avec AWS Proton, veuillez consulter [Fonctionnement de AWS Proton avec IAM](#).

Administrateur IAM : si vous êtes un administrateur IAM, vous souhaitez peut-être en savoir plus sur la façon d'écrire des politiques pour gérer l'accès à AWS Proton. Pour voir des exemples de politiques AWS Proton basées sur l'identité que vous pouvez utiliser dans IAM, veuillez consulter [Exemples de politiques basées sur l'identité pour AWS Proton](#).

Authentification par des identités

L'authentification correspond au processus par lequel vous vous connectez à AWS avec vos informations d'identification. Vous devez vous authentifier (être connecté à AWS) en tant qu'utilisateur racine d'un compte AWS, en tant qu'utilisateur IAM ou en endossant un rôle IAM.

Vous pouvez vous connecter à AWS en tant qu'identité fédérée à l'aide des informations d'identification fournies par le biais d'une source d'identité. AWS IAM Identity Center Les utilisateurs (IAM Identity Center), l'authentification unique de votre entreprise et vos informations d'identification Google ou Facebook sont des exemples d'identités fédérées. Lorsque vous vous connectez avec une identité fédérée, votre administrateur aura précédemment configuré une fédération d'identités avec des rôles IAM. Lorsque vous accédez à AWS en utilisant la fédération, vous endossez indirectement un rôle.

Selon le type d'utilisateur que vous êtes, vous pouvez vous connecter à la AWS Management Console ou au portail d'accès AWS. Pour plus d'informations sur la connexion à AWS, consultez [Connexion à votre Compte AWS](#) dans le Guide de l'utilisateur Connexion à AWS.

Si vous accédez à AWS par programmation, AWS fournit un kit de développement logiciel (SDK) et une interface de ligne de commande (CLI) pour signer cryptographiquement vos demandes en utilisant vos informations d'identification. Si vous n'utilisez pas les outils AWS, vous devez signer les requêtes vous-même. Pour plus d'informations sur l'utilisation de la méthode recommandée pour signer des demandes vous-même, consultez [Signature des demandes d'API AWS](#) dans le Guide de l'utilisateur IAM.

Quelle que soit la méthode d'authentification que vous utilisez, vous devrez peut-être fournir des informations de sécurité supplémentaires. Par exemple, AWS vous recommande d'utiliser

l'authentification multifactorielle (MFA) pour améliorer la sécurité de votre compte. Pour en savoir plus, consultez [Authentification multifactorielle](#) dans le Guide de l'utilisateur AWS IAM Identity Center et [Utilisation de l'authentification multifactorielle \(MFA\) dans l'interface AWS](#) dans le Guide de l'utilisateur IAM.

Utilisateur root Compte AWS

Lorsque vous créez un Compte AWS, vous commencez avec une seule identité de connexion disposant d'un accès complet à tous les Services AWS et ressources du compte. Cette identité est appelée utilisateur root du Compte AWS. Vous pouvez y accéder en vous connectant à l'aide de l'adresse électronique et du mot de passe que vous avez utilisés pour créer le compte. Il est vivement recommandé de ne pas utiliser l'utilisateur root pour vos tâches quotidiennes. Protégez vos informations d'identification d'utilisateur root et utilisez-les pour effectuer les tâches que seul l'utilisateur root peut effectuer. Pour obtenir la liste complète des tâches qui vous imposent de vous connecter en tant qu'utilisateur root, consultez [Tâches nécessitant des informations d'identification d'utilisateur root](#) dans le Guide de l'utilisateur IAM.

Identité fédérée

Demandez aux utilisateurs humains, et notamment aux utilisateurs qui nécessitent un accès administrateur, d'appliquer la bonne pratique consistant à utiliser une fédération avec fournisseur d'identité pour accéder à Services AWS en utilisant des informations d'identification temporaires.

Une identité fédérée est un utilisateur de l'annuaire des utilisateurs de votre entreprise, un fournisseur d'identité Web, l'AWS Directory Service, l'annuaire Identity Center ou tout utilisateur qui accède à Services AWS en utilisant des informations d'identification fournies via une source d'identité. Quand des identités fédérées accèdent à Comptes AWS, elles endossent des rôles, ces derniers fournissant des informations d'identification temporaires.

Pour une gestion des accès centralisée, nous vous recommandons d'utiliser AWS IAM Identity Center. Vous pouvez créer des utilisateurs et des groupes dans IAM Identity Center, ou vous connecter et vous synchroniser avec un ensemble d'utilisateurs et de groupes dans votre propre source d'identité pour une utilisation sur l'ensemble de vos applications et de vos Comptes AWS. Pour obtenir des informations sur IAM Identity Center, consultez [Qu'est-ce que IAM Identity Center ?](#) dans le Guide de l'utilisateur AWS IAM Identity Center.

Utilisateurs et groupes IAM

Un [utilisateur IAM](#) est une identité dans votre Compte AWS qui dispose d'autorisations spécifiques pour une seule personne ou application. Dans la mesure du possible, nous vous recommandons de

vous appuyer sur des informations d'identification temporaires plutôt que de créer des utilisateurs IAM ayant des informations d'identification à long terme tels que les clés d'accès. Toutefois, si certains cas d'utilisation spécifiques nécessitent des informations d'identification à long terme avec les utilisateurs IAM, nous vous recommandons de faire pivoter les clés d'accès. Pour plus d'informations, consultez [Rotation régulière des clés d'accès pour les cas d'utilisation nécessitant des informations d'identification](#) dans le Guide de l'utilisateur IAM.

Un [groupe IAM](#) est une identité qui concerne un ensemble d'utilisateurs IAM. Vous ne pouvez pas vous connecter en tant que groupe. Vous pouvez utiliser les groupes pour spécifier des autorisations pour plusieurs utilisateurs à la fois. Les groupes permettent de gérer plus facilement les autorisations pour de grands ensembles d'utilisateurs. Par exemple, vous pouvez avoir un groupe nommé IAMAdmins et accorder à ce groupe les autorisations d'administrer des ressources IAM.

Les utilisateurs sont différents des rôles. Un utilisateur est associé de manière unique à une personne ou une application, alors qu'un rôle est conçu pour être endossé par tout utilisateur qui en a besoin. Les utilisateurs disposent d'informations d'identification permanentes, mais les rôles fournissent des informations d'identification temporaires. Pour en savoir plus, consultez [Quand créer un utilisateur IAM \(au lieu d'un rôle\)](#) dans le Guide de l'utilisateur IAM.

Rôles IAM

Un [rôle IAM](#) est une entité au sein de votre Compte AWS qui dispose d'autorisations spécifiques. Le concept ressemble à celui d'utilisateur IAM, mais le rôle IAM n'est pas associé à une personne en particulier. Vous pouvez temporairement endosser un rôle IAM dans la AWS Management Console en [changeant de rôle](#). Vous pouvez obtenir un rôle en appelant une opération d'API AWS CLI ou AWS à l'aide d'une URL personnalisée. Pour plus d'informations sur les méthodes d'utilisation des rôles, consultez [Utilisation de rôles IAM](#) dans le Guide de l'utilisateur IAM.

Les rôles IAM avec des informations d'identification temporaires sont utiles dans les cas suivants :

- Accès utilisateur fédéré – Pour attribuer des autorisations à une identité fédérée, vous créez un rôle et définissez des autorisations pour le rôle. Quand une identité externe s'authentifie, l'identité est associée au rôle et reçoit les autorisations qui sont définies par celui-ci. Pour obtenir des informations sur les rôles pour la fédération, consultez [Création d'un rôle pour un fournisseur d'identité tiers \(fédération\)](#) dans le Guide de l'utilisateur IAM. Si vous utilisez IAM Identity Center, vous configurez un jeu d'autorisations. IAM Identity Center met en corrélation le jeu d'autorisations avec un rôle dans IAM afin de contrôler à quoi vos identités peuvent accéder après leur authentification. Pour plus d'informations sur les jeux d'autorisations, consultez [Jeux d'autorisations](#) dans le Guide de l'utilisateur AWS IAM Identity Center.

- Autorisations d'utilisateur IAM temporaires : un rôle ou un utilisateur IAM peut endosser un rôle IAM pour profiter temporairement d'autorisations différentes pour une tâche spécifique.
- Accès intercompte : vous pouvez utiliser un rôle IAM pour permettre à un utilisateur (principal de confiance) d'un compte différent d'accéder aux ressources de votre compte. Les rôles constituent le principal moyen d'accorder l'accès intercompte. Toutefois, certains Services AWS vous permettent d'attacher une politique directement à une ressource (au lieu d'utiliser un rôle en tant que proxy). Pour en savoir plus sur la différence entre les rôles et les politiques basées sur les ressources pour l'accès intercompte, consultez [Différence entre les rôles IAM et les politiques basées sur les ressources](#) dans le Guide de l'utilisateur IAM.
- Accès interservices : certains Services AWS utilisent des fonctionnalités dans d'autres Services AWS. Par exemple, lorsque vous effectuez un appel dans un service, il est courant que ce service exécute des applications dans Amazon EC2 ou stocke des objets dans Amazon S3. Un service peut le faire en utilisant les autorisations d'appel du principal, une fonction de service ou un rôle lié au service.
 - Sessions de transmission d'accès (FAS) : lorsque vous vous servez d'un utilisateur ou d'un rôle IAM pour accomplir des actions dans AWS, vous êtes considéré comme un principal. Lorsque vous utilisez certains services, vous pouvez effectuer une action qui initie une autre action dans un autre service. FAS utilise les autorisations du principal appelant un Service AWS, associées au Service AWS demandeur pour adresser des demandes aux services situés en aval. Les demandes FAS ne sont formulées que lorsqu'un service reçoit une demande qui, pour aboutir, a besoin d'interagir avec d'autres ressources ou Services AWS. Dans ce cas, vous devez disposer d'autorisations nécessaires pour effectuer les deux actions. Pour plus de détails sur la politique relative à la transmission de demandes FAS, consultez la section [Sessions de transmission d'accès](#).
- Fonction du service : il s'agit d'un [rôle IAM](#) attribué à un service afin de réaliser des actions en votre nom. Un administrateur IAM peut créer, modifier et supprimer une fonction du service à partir d'IAM. Pour plus d'informations, consultez [Création d'un rôle pour la délégation d'autorisations à un Service AWS](#) dans le Guide de l'utilisateur IAM.
- Rôle lié au service – Un rôle lié au service est un type de fonction du service lié à un Service AWS. Le service peut endosser le rôle afin d'effectuer une action en votre nom. Les rôles liés à un service s'affichent dans votre Compte AWS et sont détenus par le service. Un administrateur IAM peut consulter, mais ne peut pas modifier, les autorisations concernant les rôles liés à un service.
- Applications s'exécutant sur Amazon EC2 : vous pouvez utiliser un rôle IAM pour gérer des informations d'identification temporaires pour les applications s'exécutant sur une instance EC2

et effectuant des demandes d'API AWS CLI ou AWS. Cette solution est préférable au stockage des clés d'accès au sein de l'instance EC2. Pour attribuer un rôle AWS à une instance EC2 et le rendre disponible à toutes les applications associées, vous pouvez créer un profil d'instance attaché à l'instance. Un profil d'instance contient le rôle et permet aux programmes qui s'exécutent sur l'instance EC2 d'obtenir des informations d'identification temporaires. Pour plus d'informations, consultez [Utilisation d'un rôle IAM pour accorder des autorisations à des applications s'exécutant sur des instances Amazon EC2](#) dans le Guide de l'utilisateur IAM.

Pour savoir dans quel cas utiliser des rôles ou des utilisateurs IAM, consultez [Quand créer un rôle IAM \(au lieu d'un utilisateur\)](#) dans le Guide de l'utilisateur IAM.

Gestion des accès à l'aide de politiques

Vous contrôlez les accès dans AWS en créant des politiques et en les attachant à des identités AWS ou à des ressources. Une politique est un objet dans AWS qui, lorsqu'il est associé à une identité ou à une ressource, définit les autorisations de ces dernières. AWS évalue ces politiques lorsqu'un principal (utilisateur, utilisateur root ou séance de rôle) envoie une demande. Les autorisations dans les politiques déterminent si la demande est autorisée ou refusée. La plupart des politiques sont stockées dans AWS en tant que documents JSON. Pour plus d'informations sur la structure et le contenu des documents de politique JSON, consultez [Présentation des politiques JSON](#) dans le Guide de l'utilisateur IAM.

Les administrateurs peuvent utiliser les politiques JSON AWS pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

Par défaut, les utilisateurs et les rôles ne disposent d'aucune autorisation. Pour octroyer aux utilisateurs des autorisations d'effectuer des actions sur les ressources dont ils ont besoin, un administrateur IAM peut créer des politiques IAM. L'administrateur peut ensuite ajouter les politiques IAM aux rôles et les utilisateurs peuvent assumer les rôles.

Les politiques IAM définissent les autorisations d'une action, quelle que soit la méthode que vous utilisez pour exécuter l'opération. Par exemple, supposons que vous disposiez d'une politique qui autorise l'action `iam:GetRole`. Un utilisateur avec cette politique peut obtenir des informations utilisateur à partir de la AWS Management Console, de la AWS CLI ou de l'API AWS.

Politiques basées sur l'identité

Les politiques basées sur l'identité sont des documents de politique d'autorisations JSON que vous pouvez attacher à une identité telle qu'un utilisateur, un groupe d'utilisateurs ou un rôle IAM. Ces politiques contrôlent quel type d'actions des utilisateurs et des rôles peuvent exécuter, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l'identité, consultez [Création de politiques IAM](#) dans le Guide de l'utilisateur IAM.

Les politiques basées sur l'identité peuvent être classées comme des politiques en ligne ou des politiques gérées. Les politiques en ligne sont intégrées directement à un utilisateur, groupe ou rôle. Les politiques gérées sont des politiques autonomes que vous pouvez attacher à plusieurs utilisateurs, groupes et rôles dans votre Compte AWS. Les politiques gérées incluent les politiques gérées par AWS et les politiques gérées par le client. Pour découvrir comment choisir entre une politique gérée et une politique en ligne, consultez [Choix entre les politiques gérées et les politiques en ligne](#) dans le Guide de l'utilisateur IAM.

politiques basées sur les ressources

Les politiques basées sur les ressources sont des documents de politique JSON que vous attachez à une ressource. Des politiques basées sur les ressources sont, par exemple, les politiques de confiance de rôle IAM et des politiques de compartiment Amazon S3. Dans les services qui sont compatibles avec les politiques basées sur les ressources, les administrateurs de service peuvent les utiliser pour contrôler l'accès à une ressource spécifique. Pour la ressource dans laquelle se trouve la politique, cette dernière définit quel type d'actions un principal spécifié peut effectuer sur cette ressource et dans quelles conditions. Vous devez [spécifier un principal](#) dans une politique basée sur les ressources. Les principaux peuvent inclure des comptes, des utilisateurs, des rôles, des utilisateurs fédérés ou des Services AWS.

Les politiques basées sur les ressources sont des politiques en ligne situées dans ce service. Vous ne pouvez pas utiliser les politiques gérées AWS depuis IAM dans une politique basée sur une ressource.

Listes de contrôle d'accès (ACL)

Les listes de contrôle d'accès (ACL) vérifie quels principaux (membres de compte, utilisateurs ou rôles) ont l'autorisation d'accéder à une ressource. Les listes de contrôle d'accès sont similaires aux politiques basées sur les ressources, bien qu'elles n'utilisent pas le format de document de politique JSON.

Amazon S3, AWS WAF et Amazon VPC sont des exemples de services prenant en charge les ACL. Pour en savoir plus sur les listes de contrôle d'accès, consultez [Présentation des listes de contrôle d'accès \(ACL\)](#) dans le Guide du développeur Amazon Simple Storage Service.

Autres types de politique

AWS prend en charge d'autres types de politiques moins courantes. Ces types de politiques peuvent définir le nombre maximum d'autorisations qui vous sont accordées par des types de politiques plus courants.

- **Limite d'autorisations** : une limite d'autorisations est une fonctionnalité avancée dans laquelle vous définissez le nombre maximal d'autorisations qu'une politique basée sur l'identité peut accorder à une entité IAM (utilisateur ou rôle IAM). Vous pouvez définir une limite d'autorisations pour une entité. Les autorisations qui en résultent représentent la combinaison des politiques basées sur l'identité d'une entité et de ses limites d'autorisation. Les politiques basées sur les ressources qui spécifient l'utilisateur ou le rôle dans le champ `Principal` ne sont pas limitées par les limites d'autorisations. Un refus explicite dans l'une de ces politiques remplace l'autorisation. Pour plus d'informations sur les limites d'autorisations, consultez [Limites d'autorisations pour des entités IAM](#) dans le Guide de l'utilisateur IAM.
- **Politiques de contrôle des services (SCP)** - les SCP sont des politiques JSON qui spécifient le nombre maximal d'autorisations pour une organisation ou une unité d'organisation (OU) dans AWS Organizations. AWS Organizations est un service qui vous permet de regrouper et de gérer de façon centralisée plusieurs Comptes AWS détenus par votre entreprise. Si vous activez toutes les fonctionnalités d'une organisation, vous pouvez appliquer les politiques de contrôle des services (SCP) à l'un ou à l'ensemble de vos comptes. La SCP limite les autorisations pour les entités dans les comptes membres, y compris dans chaque Utilisateur racine d'un compte AWS. Pour plus d'informations sur les organisations et les SCP, consultez [Fonctionnement des SCP](#) dans le Guide de l'utilisateur AWS Organizations.
- **politiques de séance** : les politiques de séance sont des politiques avancées que vous utilisez en tant que paramètre lorsque vous créez par programmation une séance temporaire pour un rôle ou un utilisateur fédéré. Les autorisations de la séance obtenue sont une combinaison des politiques basées sur l'identité de l'utilisateur ou du rôle et des politiques de séance. Les autorisations peuvent également provenir d'une politique basée sur les ressources. Un refus explicite dans l'une de ces politiques remplace l'autorisation. Pour plus d'informations, consultez [Politiques de séance](#) dans le Guide de l'utilisateur IAM.

Plusieurs types de politique

Lorsque plusieurs types de politiques s'appliquent à la requête, les autorisations obtenues sont plus compliquées à comprendre. Pour découvrir la façon dont AWS détermine s'il convient d'autoriser une demande en présence de plusieurs types de politiques, consultez [Logique d'évaluation de politiques](#) dans le Guide de l'utilisateur IAM.

Fonctionnement de AWS Proton avec IAM

Avant d'utiliser IAM pour gérer l'accès à AWS Proton, découvrez les fonctions IAM que vous pouvez utiliser avec AWS Proton.

Fonctions IAM que vous pouvez utiliser avec AWS Proton

Fonction IAM	Prise en charge de AWS Proton
Politiques basées sur l'identité	Oui
Politiques basées sur les ressources	Non
Actions de politique	Oui
Ressources de politique	Oui
Clés de condition d'une politique	Oui
ACL	Non
ABAC (étiquettes dans les politiques)	Oui
Informations d'identification temporaires	Oui
Autorisations de principal	Oui
Fonctions du service	Oui
Rôles liés à un service	Oui

Pour obtenir une vue d'ensemble de la façon dont la plupart des fonctionnalités IAM Services AWS fonctionnent AWS Proton et des autres fonctionnalités, consultez Services AWS le [guide de l'utilisateur IAM](#) consacré à leur utilisation.

Politiques basées sur l'identité pour AWS Proton

Prend en charge les politiques basées sur une identité Oui

Les politiques basées sur l'identité sont des documents de politique d'autorisations JSON que vous pouvez attacher à une identité telle qu'un utilisateur, un Groupes d'utilisateurs IAM ou un rôle IAM. Ces politiques contrôlent quel type d'actions des utilisateurs et des rôles peuvent exécuter, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l'identité, consultez [Création de politiques IAM](#) dans le Guide de l'utilisateur IAM.

Avec les politiques IAM basées sur l'identité, vous pouvez spécifier des actions et ressources autorisées ou refusées, ainsi que les conditions dans lesquelles les actions sont autorisées ou refusées. Vous ne pouvez pas spécifier le principal dans une politique basée sur une identité car celle-ci s'applique à l'utilisateur ou au rôle auquel elle est attachée. Pour découvrir tous les éléments que vous utilisez dans une politique JSON, consultez [Références des éléments de politique JSON IAM](#) dans le Guide de l'utilisateur IAM.

Exemples de politiques basées sur l'identité pour AWS Proton

Pour voir des exemples de politiques AWS Proton basées sur l'identité, consultez [Exemples de politiques basées sur l'identité pour AWS Proton](#).

Politiques basées sur une ressource dans AWS Proton

Prend en charge les politiques basées sur une ressource Non

Les politiques basées sur les ressources sont des documents de politique JSON que vous attachez à une ressource. Des politiques basées sur les ressources sont, par exemple, les politiques de confiance de rôle IAM et des politiques de compartiment Amazon S3. Dans les services qui sont compatibles avec les politiques basées sur les ressources, les administrateurs de service peuvent les

utiliser pour contrôler l'accès à une ressource spécifique. Pour la ressource dans laquelle se trouve la politique, cette dernière définit quel type d'actions un principal spécifié peut effectuer sur cette ressource et dans quelles conditions. Vous devez [spécifier un principal](#) dans une politique basée sur les ressources. Les principaux peuvent inclure des comptes, des utilisateurs, des rôles, des utilisateurs fédérés ou des Services AWS.

Pour permettre un accès intercompte, vous pouvez spécifier un compte entier ou des entités IAM dans un autre compte en tant que principal dans une politique basée sur les ressources. L'ajout d'un principal entre comptes à une politique basée sur les ressources ne représente qu'une partie de l'instauration de la relation d'approbation. Quand le principal et la ressource se trouvent dans des Comptes AWS différents, un administrateur IAM dans le compte approuvé doit également accorder à l'entité principal (utilisateur ou rôle) l'autorisation d'accéder à la ressource. Pour ce faire, il attache une politique basée sur une identité à l'entité. Toutefois, si une politique basée sur des ressources accorde l'accès à un principal dans le même compte, aucune autre politique basée sur l'identité n'est requise. Pour plus d'informations, consultez [Différence entre les rôles IAM et les politiques basées sur une ressource](#) dans le Guide de l'utilisateur IAM.

Actions de stratégie pour AWS Proton

Prend en charge les actions de politique	Oui
--	-----

Les administrateurs peuvent utiliser les politiques JSON AWS pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément `Action` d'une politique JSON décrit les actions que vous pouvez utiliser pour autoriser ou refuser l'accès à une politique. Les actions de politique possèdent généralement le même nom que l'opération d'API AWS associée. Il existe quelques exceptions, telles que les actions avec autorisations uniquement qui n'ont pas d'opération API correspondante. Certaines opérations nécessitent également plusieurs actions dans une politique. Ces actions supplémentaires sont nommées actions dépendantes.

Intégration d'actions dans une politique afin d'accorder l'autorisation d'exécuter les opérations associées.

Pour afficher la liste des actions AWS Proton, consultez [Actions définies par AWS Proton](#) dans la Référence de l'autorisation de service.

Les actions de politique dans AWS Proton utilisent le préfixe suivant avant l'action : .

```
proton
```

Pour indiquer plusieurs actions dans une seule déclaration, séparez-les par des virgules.

```
"Action": [  
  "proton:action1",  
  "proton:action2"  
]
```

Vous pouvez aussi préciser plusieurs actions à l'aide de caractères génériques (*). Par exemple, pour spécifier toutes les actions qui commencent par le mot `List`, incluez l'action suivante :

```
"Action": "proton:List*"
```

Pour voir des exemples de politiques AWS Proton basées sur l'identité, consultez [Exemples de politiques basées sur l'identité pour AWS Proton](#).

Ressources de politique pour AWS Proton

Prend en charge les ressources de politique	Oui
---	-----

Les administrateurs peuvent utiliser les politiques JSON AWS pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément de politique JSON `Resource` indique le ou les objets auxquels l'action s'applique. Les instructions doivent inclure un élément `Resource` ou `NotResource`. Il est recommandé de définir une ressource à l'aide de son [Amazon Resource Name \(ARN\)](#). Vous pouvez le faire pour des actions qui prennent en charge un type de ressource spécifique, connu sous la dénomination autorisations de niveau ressource.

Pour les actions qui ne sont pas compatibles avec les autorisations de niveau ressource, telles que les opérations de liste, utilisez un caractère générique (*) afin d'indiquer que l'instruction s'applique à toutes les ressources.

```
"Resource": "*"
```

Pour afficher la liste des types de ressources AWS Proton et leurs ARN, consultez [Ressources définies par AWS Proton](#) dans la Référence de l'autorisation de service. Pour savoir grâce à quelles actions vous pouvez spécifier l'ARN de chaque ressource, consultez [Actions définies par AWS Proton](#).

Pour voir des exemples de politiques AWS Proton basées sur l'identité, consultez [Exemples de politiques basées sur l'identité pour AWS Proton](#).

Clés de condition de politique pour AWS Proton

Prise en charge des clés de condition de stratégie spécifiques au service	Oui
---	-----

Les administrateurs peuvent utiliser les politiques JSON AWS pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément `Condition` (ou le bloc `Condition`) vous permet de spécifier des conditions lorsqu'une instruction est appliquée. L'élément `Condition` est facultatif. Vous pouvez créer des expressions conditionnelles qui utilisent des [opérateurs de condition](#), tels que les signes égal ou inférieur à, pour faire correspondre la condition de la politique aux valeurs de la demande.

Si vous spécifiez plusieurs éléments `Condition` dans une instruction, ou plusieurs clés dans un seul élément `Condition`, AWS les évalue à l'aide d'une opération AND logique. Si vous spécifiez plusieurs valeurs pour une seule clé de condition, AWS évalue la condition à l'aide d'une opération OR logique. Toutes les conditions doivent être remplies avant que les autorisations associées à l'instruction ne soient accordées.

Vous pouvez aussi utiliser des variables d'espace réservé quand vous spécifiez des conditions. Par exemple, vous pouvez accorder à un utilisateur IAM l'autorisation d'accéder à une ressource uniquement si elle est balisée avec son nom d'utilisateur IAM. Pour plus d'informations, consultez [Éléments d'une politique IAM : variables et identifications](#) dans le Guide de l'utilisateur IAM.

AWS prend en charge les clés de condition globales et les clés de condition spécifiques à un service. Pour afficher toutes les clés de condition globales AWS, consultez [Clés de contexte de condition globale AWS](#) dans le Guide de l'utilisateur IAM.

Pour afficher la liste des clés de condition AWS Proton, consultez [Clés de condition pour AWS Proton](#) dans la Référence de l'autorisation de service. Pour savoir avec quelles actions et ressources vous pouvez utiliser une clé de condition, consultez [Actions définies par AWS Proton](#).

Pour consulter un exemple condition-key-based de politique visant à limiter l'accès à une ressource, consultez [Exemples de politiques basées sur des clés conditionnelles pour AWS Proton](#).

Listes de contrôle d'accès (ACL) dans AWS Proton

Prend en charge les listes ACL	Non
--------------------------------	-----

Les listes de contrôle d'accès (ACL) vérifient quels principaux (membres de compte, utilisateurs ou rôles) ont l'autorisation d'accéder à une ressource. Les listes de contrôle d'accès sont similaires aux politiques basées sur les ressources, bien qu'elles n'utilisent pas le format de document de politique JSON.

Les listes de contrôle d'accès (ACL) sont des listes de bénéficiaires que vous pouvez attacher aux ressources. Elles accordent des autorisations aux comptes pour accéder aux ressources auxquelles ils sont associés.

Contrôle d'accès par attributs (ABAC) avec AWS Proton

Prend en charge ABAC (étiquettes dans les politiques)	Oui
---	-----

Le contrôle d'accès par attributs (ABAC) est une stratégie d'autorisation qui définit des autorisations en fonction des attributs. Dans AWS, ces attributs sont appelés étiquettes. Vous pouvez attacher des étiquettes à des entités IAM (utilisateurs ou rôles), ainsi qu'à de nombreuses ressources AWS. L'étiquetage des entités et des ressources est la première étape d'ABAC. Vous concevez ensuite des politiques ABAC pour autoriser des opérations quand l'identification du principal correspond à celle de la ressource à laquelle il tente d'accéder.

L'ABAC est utile dans les environnements qui connaissent une croissance rapide et pour les cas où la gestion des politiques devient fastidieuse.

Pour contrôler l'accès basé sur des balises, vous devez fournir les informations de balise dans l'[élément de condition](#) d'une politique utilisant les clés de condition `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` ou `aws:TagKeys`.

Si un service prend en charge les trois clés de condition pour tous les types de ressources, alors la valeur pour ce service est Oui. Si un service prend en charge les trois clés de condition pour certains types de ressources uniquement, la valeur est Partielle.

Pour plus d'informations sur l'ABAC, consultez [Qu'est-ce que le contrôle d'accès basé sur les attributs \(ABAC\) ?](#) dans le Guide de l'utilisateur IAM. Pour accéder à un didacticiel décrivant les étapes de configuration de l'ABAC, consultez [Utilisation du contrôle d'accès basé sur les attributs \(ABAC\)](#) dans le Guide de l'utilisateur IAM.

Pour plus d'informations sur le balisage des ressources AWS Proton, consultez [AWS Protonressources et balises](#).

Utilisation d'informations d'identification temporaires avec AWS Proton

Prend en charge les informations d'identification temporaires	Oui
---	-----

Certains Services AWS ne fonctionnent pas quand vous vous connectez à l'aide d'informations d'identification temporaires. Pour plus d'informations, notamment sur les Services AWS qui fonctionnent avec des informations d'identification temporaires, consultez [Services AWS qui fonctionnent avec IAM](#) dans le Guide de l'utilisateur IAM.

Vous utilisez des informations d'identification temporaires quand vous vous connectez à la AWS Management Console en utilisant toute méthode autre qu'un nom d'utilisateur et un mot de passe. Par exemple, lorsque vous accédez à AWS en utilisant le lien d'authentification unique (SSO) de votre société, ce processus crée automatiquement des informations d'identification temporaires. Vous créez également automatiquement des informations d'identification temporaires lorsque vous vous connectez à la console en tant qu'utilisateur, puis changez de rôle. Pour plus d'informations sur le changement de rôle, consultez [Changement de rôle \(console\)](#) dans le Guide de l'utilisateur IAM.

Vous pouvez créer manuellement des informations d'identification temporaires à l'aide d'AWS CLI ou de l'API AWS. Vous pouvez ensuite utiliser ces informations d'identification temporaires pour

accéder à AWS. AWS recommande de générer des informations d'identification temporaires de façon dynamique au lieu d'utiliser des clés d'accès à long terme. Pour plus d'informations, consultez [Informations d'identification de sécurité temporaires dans IAM](#).

Autorisations principales entre services pour AWS Proton

Prend en charge les transmissions de sessions d'accès (FAS) Oui

Lorsque vous vous servez d'un utilisateur IAM ou d'un rôle IAM pour accomplir des actions dans AWS, vous êtes considéré comme un principal. Lorsque vous utilisez certains services, l'action que vous effectuez est susceptible de lancer une autre action dans un autre service. FAS utilise les autorisations du principal appelant un Service AWS, associées au Service AWS demandeur pour adresser des demandes aux services situés en aval. Les demandes FAS ne sont formulées que lorsqu'un service reçoit une demande qui, pour aboutir, a besoin d'interagir avec d'autres ressources ou Services AWS. Dans ce cas, vous devez disposer d'autorisations nécessaires pour effectuer les deux actions. Pour plus de détails sur la politique relative à la transmission de demandes FAS, consultez la section [Sessions de transmission d'accès](#).

Fonctions de service pour AWS Proton

Prend en charge les fonctions du service Oui

Une fonction du service est un [rôle IAM](#) qu'un service endosse pour accomplir des actions en votre nom. Un administrateur IAM peut créer, modifier et supprimer une fonction du service à partir d'IAM. Pour plus d'informations, consultez [Création d'un rôle pour la délégation d'autorisations à un Service AWS](#) dans le Guide de l'utilisateur IAM.

Pour plus d'informations, consultez [AWS Proton Exemples de politiques relatives aux rôles de service IAM](#).

Warning

La modification des autorisations d'une fonction du service peut altérer la fonctionnalité d'AWS Proton. Ne modifiez des fonctions du service que quand AWS Proton vous le conseille.

Rôles liés à un service pour AWS Proton

Prend en charge les rôles liés à un service. Oui

Un rôle lié à un service est un type de fonction du service lié à un Service AWS. Le service peut endosser le rôle afin d'effectuer une action en votre nom. Les rôles liés à un service s'affichent dans votre Compte AWS et sont détenus par le service. Un administrateur IAM peut consulter, mais ne peut pas modifier, les autorisations concernant les rôles liés à un service.

Pour plus de détails sur la création ou la gestion des rôles liés à un service, consultez la section relative à l'[Services AWS Utilisation d'IAM](#). Recherchez un service dans le tableau qui inclut un Yes dans la colonne Rôle lié à un service. Choisissez le lien Oui pour consulter la documentation du rôle lié à ce service.

Exemples de politiques pour AWS Proton

Vous trouverez des exemples de politiques AWS Proton IAM dans les sections suivantes.

Rubriques

- [Exemples de politiques basées sur l'identité pour AWS Proton](#)
- [AWS Proton Exemples de politiques relatives aux rôles de service IAM](#)
- [Exemples de politiques basées sur des clés conditionnelles pour AWS Proton](#)

Exemples de politiques basées sur l'identité pour AWS Proton

Par défaut, les utilisateurs et les rôles ne sont pas autorisés à créer ou modifier les ressources AWS Proton. Ils ne peuvent pas non plus exécuter des tâches à l'aide de la AWS Management Console, de l'AWS Command Line Interface (AWS CLI) ou de l'API AWS. Pour octroyer aux utilisateurs des autorisations d'effectuer des actions sur les ressources dont ils ont besoin, un administrateur IAM doit créer des politiques IAM. L'administrateur peut ensuite ajouter les politiques IAM aux rôles et les utilisateurs peuvent assumer les rôles.

Pour apprendre à créer une politique basée sur l'identité IAM à l'aide de ces exemples de documents de politique JSON, consultez [Création de politiques dans l'onglet JSON](#) dans le Guide de l'utilisateur IAM.

Pour plus de détails sur les actions et les types de ressources définis par AWS Proton, y compris le format des ARN pour chacun des types de ressources, consultez [Actions, ressources et clés de condition pour AWS Proton](#) dans la Référence de l'autorisation de service.

Rubriques

- [Bonnes pratiques en matière de politiques](#)
- [Liens vers des exemples de politiques basées sur l'identité pour AWS Proton](#)

Bonnes pratiques en matière de politiques

Les stratégies basées sur l'identité déterminent si une personne peut créer, consulter ou supprimer des ressources AWS Proton dans votre compte. Ces actions peuvent entraîner des frais pour votre Compte AWS. Lorsque vous créez ou modifiez des politiques basées sur l'identité, suivez ces instructions et recommandations :

- Démarrer avec AWS gérées et évoluez vers les autorisations de moindre privilège - Pour commencer à accorder des autorisations à vos utilisateurs et charges de travail, utilisez les politiques gérées AWS qui accordent des autorisations dans de nombreux cas d'utilisation courants. Elles sont disponibles dans votre Compte AWS. Nous vous recommandons de réduire encore les autorisations en définissant des politiques gérées par le client AWS qui sont spécifiques à vos cas d'utilisation. Pour plus d'informations, consultez [Politiques gérées AWS](#) ou [Politiques gérées AWS pour les activités professionnelles](#) dans le Guide de l'utilisateur IAM.
- Accorder les autorisations de moindre privilège - Lorsque vous définissez des autorisations avec des politiques IAM, accordez uniquement les autorisations nécessaires à l'exécution d'une seule tâche. Pour ce faire, vous définissez les actions qui peuvent être entreprises sur des ressources spécifiques dans des conditions spécifiques, également appelées autorisations de moindre privilège. Pour plus d'informations sur l'utilisation d'IAM pour appliquer des autorisations, consultez [Politiques et autorisations dans IAM](#) dans le Guide de l'utilisateur IAM.
- Utiliser des conditions dans les politiques IAM pour restreindre davantage l'accès - Vous pouvez ajouter une condition à vos politiques afin de limiter l'accès aux actions et aux ressources. Par exemple, vous pouvez écrire une condition de politique pour spécifier que toutes les demandes doivent être envoyées via SSL. Vous pouvez également utiliser des conditions pour accorder l'accès aux actions de service si elles sont utilisées via un Service AWS spécifique, comme AWS CloudFormation. Pour plus d'informations, consultez [Conditions pour éléments de politique JSON IAM](#) dans le Guide de l'utilisateur IAM.

- Utilisez IAM Access Analyzer pour valider vos politiques IAM afin de garantir des autorisations sécurisées et fonctionnelles - IAM Access Analyzer valide les politiques nouvelles et existantes de manière à ce que les politiques IAM respectent le langage de politique IAM (JSON) et les bonnes pratiques IAM. IAM Access Analyzer fournit plus de 100 vérifications de politiques et des recommandations exploitables pour vous aider à créer des politiques sécurisées et fonctionnelles. Pour plus d'informations, consultez [Validation de politique IAM Access Analyzer](#) dans le Guide de l'utilisateur IAM.
- Authentification multifactorielle (MFA) nécessaire : si vous avez un scénario qui nécessite des utilisateurs IAM ou un utilisateur root dans votre Compte AWS, activez l'authentification multifactorielle pour une sécurité renforcée. Pour exiger le MFA lorsque des opérations d'API sont appelées, ajoutez des conditions MFA à vos politiques. Pour plus d'informations, consultez [Configuration de l'accès aux API protégé par MFA](#) dans le Guide de l'utilisateur IAM.

Pour plus d'informations sur les bonnes pratiques dans IAM, consultez [Bonnes pratiques de sécurité dans IAM](#) dans le Guide de l'utilisateur IAM.

Liens vers des exemples de politiques basées sur l'identité pour AWS Proton

Liens vers des exemples de politiques basées sur l'identité pour AWS Proton

- [Politiques gérées par AWS pour AWS Proton](#)
- [AWS Proton Exemples de politiques relatives aux rôles de service IAM](#)
- [Exemples de politiques basées sur des clés conditionnelles pour AWS Proton](#)

AWS Proton Exemples de politiques relatives aux rôles de service IAM

Les administrateurs possèdent et gèrent les ressources AWS Proton créées conformément aux modèles d'environnement et de service. Ils associent à leur compte des rôles de service IAM qui leur permettent AWS Proton de créer des ressources en leur nom. Les administrateurs fournissent les rôles et les AWS Key Management Service clés IAM pour les ressources qui sont ensuite détenues et gérées par les développeurs lors du AWS Proton déploiement de leur application en tant que AWS Proton service dans un AWS Proton environnement. Pour plus d'informations sur le chiffrement des données AWS KMS et le chiffrement des données, consultez [Protection des données dans AWS Proton](#).

Un rôle de service est un rôle Amazon Web Services (IAM) qui permet AWS Proton d'appeler des ressources en votre nom. Si vous spécifiez un rôle de service, AWS Proton utilise les informations

d'identification du rôle. Utilisez un rôle de service pour spécifier explicitement les actions que AWS Proton peut effectuer.

Le rôle de service et sa politique d'autorisation sont créés à partir du service IAM. Pour plus d'informations sur la création d'une fonction de service, veuillez consulter la rubrique [Création d'un rôle pour la délégation d'autorisations à un service AWS](#) dans le Guide de l'utilisateur IAM.

AWS Proton rôle de service pour le provisionnement à l'aide AWS CloudFormation

En tant que membre de l'équipe de la plateforme, vous pouvez, en tant qu'administrateur, créer un rôle de service AWS Proton et le fournir AWS Proton lorsque vous créez un environnement en tant que rôle de service de l'environnement (`protonServiceRoleArn` paramètre de l'action d'[CreateEnvironment](#) API). Ce rôle permet d'effectuer des appels d'API vers d'autres services en votre nom lorsque l'environnement ou l'une des instances de service qui y sont exécutées utilise le provisionnement AWS géré et AWS CloudFormation pour provisionner l'infrastructure.

Nous vous recommandons d'utiliser le rôle IAM et la politique de confiance suivants pour votre rôle de service AWS Proton. Lorsque vous utilisez la console AWS Proton pour créer un environnement et que vous choisissez de créer un nouveau rôle, cette politique s'ajoute au rôle de service qu'elle crée pour vous. Lorsque vous délimitez l'autorisation associée à cette politique, gardez à l'esprit qu'elle échoue en cas d'`Access Denied`.

Important

Sachez que les politiques présentées dans les exemples suivants accordent des privilèges d'administrateur à toute personne habilitée à enregistrer un modèle sur votre compte. Comme nous ne savons pas quelles ressources vous allez définir dans vos modèles AWS Proton, ces politiques comportent des autorisations étendues. Nous vous recommandons de limiter les autorisations aux ressources spécifiques qui seront déployées dans vos environnements.

AWS Proton exemple de politique de rôle de service pour AWS CloudFormation

Remplacez `123456789012` par votre ID Compte AWS.

```
{
  "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 politique de confiance en matière de services

```

{
  "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/*"
      }
    }
  }
}

```

Politique de rôle du service AWS de provisionnement géré et limitée

Voici un exemple de politique de rôle de AWS Proton service délimitée que vous pouvez utiliser si vous n'avez besoin de AWS Proton services que pour provisionner des ressources S3.

```

{
  "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 rôle de service pour le CodeBuild provisionnement

En tant que membre de l'équipe de la plateforme, vous pouvez, en tant qu'administrateur, créer un rôle de AWS Proton service et le fournir AWS Proton lorsque vous créez un environnement en tant que rôle de CodeBuild service de l'environnement (codebuildRoleArn paramètre de l'action d'[CreateEnvironment](#) API). Ce rôle permet d'AWS Proton effectuer des appels d'API vers d'autres

services en votre nom lorsque l'environnement ou l'une des instances de service qui y sont exécutées utilise le CodeBuild provisionnement pour provisionner l'infrastructure.

Lorsque vous utilisez la AWS Proton console pour créer un environnement et que vous choisissez de créer un nouveau rôle, AWS Proton une politique avec des privilèges d'administrateur est ajoutée au rôle de service créé pour vous. Lorsque vous créez votre propre rôle et que vous limitez les autorisations, gardez à l'esprit que cela AWS Proton échoue en cas d'Access Denied erreur.

Important

Sachez que les politiques associées aux AWS Proton rôles qu'il crée pour vous accordent des privilèges d'administrateur à toute personne habilitée à enregistrer un modèle sur votre compte. Comme nous ne savons pas quelles ressources vous allez définir dans vos AWS Proton modèles, ces politiques comportent des autorisations étendues. Nous vous recommandons de limiter les autorisations aux ressources spécifiques qui seront déployées dans vos environnements.

AWS Proton exemple de politique de rôle de service pour CodeBuild

L'exemple suivant fournit des autorisations permettant de CodeBuild provisionner des ressources à l'aide du AWS Cloud Development Kit (AWS CDK).

Remplacez **123456789012** par votre ID Compte AWS.

```
{
  "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 politique de confiance

```

{
  "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 rôles de service de pipeline

Pour provisionner des pipelines de services, vous AWS Proton devez disposer d'autorisations pour effectuer des appels d'API vers d'autres services. Les rôles de service requis sont similaires à ceux

que vous fournissez lorsque vous créez des environnements. Cependant, les rôles de création de pipelines sont partagés entre tous les services de votre AWS compte, et vous fournissez ces rôles sous forme de paramètres de compte dans la console ou via l'action d'[UpdateAccountSettingsAPI](#).

Lorsque vous utilisez la AWS Proton console pour mettre à jour les paramètres du compte et que vous choisissez de créer un nouveau rôle pour les rôles de CodeBuild service AWS CloudFormation ou pour les rôles de service, les politiques qui s'ajoutent aux rôles de service qu'elle crée pour vous sont les mêmes que celles décrites dans les sections précédentes, [AWS-rôle de provisionnement géré](#) et [CodeBuild rôle de provisionnement](#). Lorsque vous délimitez l'autorisation associée à cette politique, gardez à l'esprit qu'elle échoue en cas d'Access Denied.

Important

Sachez que les exemples de politiques présentés dans les sections précédentes accordent des privilèges d'administrateur à toute personne habilitée à enregistrer un modèle sur votre compte. Comme nous ne savons pas quelles ressources vous allez définir dans vos AWS Proton modèles, ces politiques comportent des autorisations étendues. Nous vous recommandons de limiter les autorisations aux ressources spécifiques qui seront déployées dans vos pipelines.

AWS Proton rôle du composant

En tant que membre de l'équipe de la plateforme, vous pouvez, en tant qu'administrateur, créer un rôle de AWS Proton service et le fournir AWS Proton lorsque vous créez un environnement en tant que rôle de CloudFormation composant de l'environnement (`componentRoleArn` paramètre de l'action d'[CreateEnvironmentAPI](#)). Ce rôle limite l'infrastructure que les composants directement définis peuvent fournir. Pour plus d'informations sur les composants, consultez [Composants](#).

L'exemple de politique suivant prend en charge la création d'un composant directement défini qui approvisionne un compartiment Amazon Simple Storage Service (Amazon S3) et une politique d'accès associée.

AWS Proton exemple de politique de rôle de composant

Remplacez **123456789012** par votre ID Compte AWS.

```
{  
  "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 politique de confiance des composants

```
{
  "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/*"
      }
    }
  }
}
```

Exemples de politiques basées sur des clés conditionnelles pour AWS Proton

L'exemple de politique IAM suivant refuse l'accès aux AWS Proton actions correspondant aux modèles spécifiés dans le Condition bloc. Notez que ces clés de condition ne sont prises en charge que par les actions répertoriées dans [Actions, ressources et clés de condition pour AWS Proton](#). Pour gérer les autorisations relatives à d'autres actions, par exemple `DeleteEnvironmentTemplate`, vous devez utiliser le contrôle d'accès au niveau des ressources.

Exemple de politique qui refuse les actions d'un AWS Proton modèle sur un modèle spécifique :

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

```

Dans l'exemple de politique suivant, la première instruction au niveau des ressources refuse l'accès aux actions du AWS Proton modèle autres que celles `ListServiceTemplates` qui correspondent au modèle de service répertorié dans le `Resource` bloc. La deuxième déclaration refuse l'accès aux AWS Proton actions correspondant au modèle répertorié dans le `Condition` bloc.

Exemple de politique refusant AWS Proton les actions correspondant à un modèle spécifique :

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "proton:*"
      ],
      "Resource": "arn:aws:region_id:123456789012:service-template/my-service-
template"
    },
    {
      "Effect": "Deny",
      "Action": [
        "proton:*"
      ],
      "Resource": "*"
    }
  ]
}

```



```

        "Condition": {
            "StringEqualsIfExists": {
                "proton:ServiceTemplate": [
                    "arn:aws:proton:region_id:123456789012:service-template/my-
service-template"
                ]
            }
        }
    ]
}

```

Le dernier exemple de politique autorise AWS Proton les actions du développeur qui correspondent au modèle de service spécifique répertorié dans le Condition bloc.

Exemple de politique permettant aux AWS Proton développeurs d'effectuer des actions correspondant à un modèle spécifique :

```

{
  "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"
            }
        }
    }
]
```

Politiques gérées par AWS pour AWS Proton

Pour ajouter des autorisations à des utilisateurs, des groupes et des rôles, il est plus facile d'utiliser des politiques gérées par AWS que d'écrire des politiques vous-même. Il faut du temps et de l'expertise pour [créer des politiques gérées par le client IAM](#) qui ne fournissent à votre équipe que les autorisations dont elle a besoin. Pour démarrer rapidement, vous pouvez utiliser nos politiques gérées par AWS. Ces politiques couvrent des cas d'utilisation courants et sont disponibles dans votre Compte AWS. Pour plus d'informations sur les politiques gérées par AWS, consultez [Politiques gérées par AWS](#) dans le Guide de l'utilisateur IAM.

Les Services AWS assurent la maintenance et la mise à jour des politiques gérées par AWS. Vous ne pouvez pas modifier les autorisations définies dans les politiques gérées par AWS. Les services ajoutent occasionnellement des autorisations à une politique gérée par AWS pour prendre en charge de nouvelles fonctionnalités. Ce type de mise à jour affecte toutes les identités (utilisateurs, groupes et rôles) auxquelles la politique est attachée. Les services sont très susceptibles de mettre à jour une politique gérée par AWS quand une nouvelle fonctionnalité est lancée ou quand de nouvelles opérations sont disponibles. Les services ne suppriment pas les autorisations d'une politique gérée par AWS, les mises à jour de politique n'interrompent vos autorisations existantes.

En outre, AWS prend en charge des politiques gérées pour des activités professionnelles couvrant plusieurs services. Par exemple, la politique `ReadOnlyAccess` gérée par AWS donne accès en lecture seule à tous les Services AWS et ressources. Quand un service lance une nouvelle fonctionnalité, AWS ajoute des autorisations en lecture seule pour les nouvelles opérations et ressources. Pour obtenir la liste des politiques de fonctions professionnelles et leurs descriptions, consultez [Politiques gérées par AWS pour les fonctions de tâches](#) dans le Guide de l'utilisateur IAM.

AWS Proton fournit des politiques IAM gérées et des relations de confiance que vous pouvez associer à des utilisateurs, à des groupes ou à des rôles qui permettent différents niveaux de contrôle sur les ressources et les opérations d'API. Vous pouvez appliquer ces politiques directement ou les utiliser comme points de départ pour créer vos propres politiques.

La relation de confiance suivante est utilisée pour chacune des politiques AWS Proton gérées.

```
{
  "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/*"
      }
    }
  }
}
```

Politique gérée par AWS : `AWSProtonFullAccess`

Vous pouvez les associer `AWSProtonFullAccess` à vos entités IAM. AWS Proton associe également cette politique à un rôle de service qui permet AWS Proton d'effectuer des actions en votre nom.

Cette politique accorde des autorisations administratives qui permettent un accès complet aux AWS Proton actions et un accès limité aux autres actions de AWS service qui en AWS Proton dépendent.

La politique inclut les espaces de noms d'actions clés suivants :

- `proton`— Permet aux administrateurs d'accéder pleinement aux AWS Proton API.
- `iam`— Permet aux administrateurs de transmettre des rôles à AWS Proton. Cela est nécessaire pour AWS Proton pouvoir effectuer des appels d'API vers d'autres services au nom de l'administrateur.
- `kms`— Permet aux administrateurs d'ajouter une autorisation à une clé gérée par le client.
- `codestar-connections`— Permet aux administrateurs de répertorier et de transmettre les connexions `codestar` afin qu'elles puissent être utilisées par AWS Proton

Détails de l'autorisation

Cette politique inclut les autorisations suivantes.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "proton:*",
        "kms:ListAliases",
        "kms:DescribeKey",
        "codestar-connections:ListConnections"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "kms:CreateGrant"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "kms:ViaService": "proton.*.amazonaws.com"
        }
      }
    }
  ],
  {
    "Effect": "Allow",
```

```

    "Action": [
      "iam:PassRole"
    ],
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "iam:PassedToService": "proton.amazonaws.com"
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": "iam:CreateServiceLinkedRole",
    "Resource": "arn:aws:iam::*:role/aws-service-role/sync.proton.amazonaws.com/AWSServiceRoleForProtonSync",
    "Condition": {
      "StringEquals": {
        "iam:AWSServiceName": "sync.proton.amazonaws.com"
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "codestar-connections:PassConnection"
    ],
    "Resource": "arn:aws:codestar-connections::*:connection/*",
    "Condition": {
      "StringEquals": {
        "codestar-connections:PassedToService": "proton.amazonaws.com"
      }
    }
  }
]
}

```

Politique gérée par AWS : AWSProtonDeveloperAccess

Vous pouvez les associer AWSProtonDeveloperAccess à vos entités IAM. AWS Proton associe également cette politique à un rôle de service qui permet AWS Proton d'effectuer des actions en votre nom.

Cette politique accorde des autorisations qui permettent un accès limité aux AWS Proton actions et aux autres AWS actions qui en AWS Proton dépendent. L'étendue de ces autorisations est conçue pour soutenir le rôle d'un développeur qui crée et déploie AWS Proton des services.

Cette politique ne donne pas accès aux API de création, de suppression et de mise à jour de AWS Proton modèles et d'environnements. Si les développeurs ont besoin d'autorisations encore plus limitées que celles prévues par cette politique, nous recommandons de créer une politique personnalisée dont l'étendue est limitée de manière à accorder le [moindre privilège](#).

La politique inclut les espaces de noms d'actions clés suivants :

- `proton`— Permet aux contributeurs d'accéder à un ensemble limité d'AWS ProtonAPI.
- `codestar-connections`— Permet aux contributeurs de répertorier et de transmettre les connexions codestar afin qu'elles puissent être utilisées par. AWS Proton

Détails de l'autorisation

Cette politique inclut les autorisations suivantes.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codecommit:ListRepositories",
        "codepipeline:GetPipeline",
        "codepipeline:GetPipelineExecution",
        "codepipeline:GetPipelineState",
        "codepipeline:ListPipelineExecutions",
        "codepipeline:ListPipelines",
        "codestar-connections:ListConnections",
        "codestar-connections:UseConnection",
        "proton:CancelServiceInstanceDeployment",
        "proton:CancelServicePipelineDeployment",
        "proton:CreateService",
        "proton>DeleteService",
        "proton:GetAccountRoles",
        "proton:GetAccountSettings",
        "proton:GetEnvironment",
        "proton:GetEnvironmentAccountConnection",
        "proton:GetEnvironmentTemplate",

```

```

    "proton:GetEnvironmentTemplateMajorVersion",
    "proton:GetEnvironmentTemplateMinorVersion",
    "proton:GetEnvironmentTemplateVersion",
    "proton:GetRepository",
    "proton:GetRepositorySyncStatus",
    "proton:GetResourcesSummary",
    "proton:GetService",
    "proton:GetServiceInstance",
    "proton:GetServiceTemplate",
    "proton:GetServiceTemplateMajorVersion",
    "proton:GetServiceTemplateMinorVersion",
    "proton:GetServiceTemplateVersion",
    "proton:GetTemplateSyncConfig",
    "proton:GetTemplateSyncStatus",
    "proton:ListEnvironmentAccountConnections",
    "proton:ListEnvironmentOutputs",
    "proton:ListEnvironmentProvisionedResources",
    "proton:ListEnvironments",
    "proton:ListEnvironmentTemplateMajorVersions",
    "proton:ListEnvironmentTemplateMinorVersions",
    "proton:ListEnvironmentTemplates",
    "proton:ListEnvironmentTemplateVersions",
    "proton:ListRepositories",
    "proton:ListRepositorySyncDefinitions",
    "proton:ListServiceInstanceOutputs",
    "proton:ListServiceInstanceProvisionedResources",
    "proton:ListServiceInstances",
    "proton:ListServicePipelineOutputs",
    "proton:ListServicePipelineProvisionedResources",
    "proton:ListServices",
    "proton:ListServiceTemplateMajorVersions",
    "proton:ListServiceTemplateMinorVersions",
    "proton:ListServiceTemplates",
    "proton:ListServiceTemplateVersions",
    "proton:ListTagsForResource",
    "proton:UpdateService",
    "proton:UpdateServiceInstance",
    "proton:UpdateServicePipeline",
    "s3:ListAllMyBuckets",
    "s3:ListBucket"
  ],
  "Resource": "*"
},
{

```

```

    "Effect": "Allow",
    "Action": "codestar-connections:PassConnection",
    "Resource": "arn:aws:codestar-connections:*:*:connection/*",
    "Condition": {
      "StringEquals": {
        "codestar-connections:PassedToService": "proton.amazonaws.com"
      }
    }
  }
]
}

```

Politique gérée par AWS : AWSProtonReadOnlyAccess

Vous pouvez les associer `AWSProtonReadOnlyAccess` à vos entités IAM. AWS Proton associe également cette politique à un rôle de service qui permet AWS Proton d'effectuer des actions en votre nom.

Cette politique accorde des autorisations qui autorisent un accès en lecture seule aux AWS Proton actions et un accès en lecture seule limité aux autres actions de AWS service qui en dépendent. AWS Proton

La politique inclut les espaces de noms d'actions clés suivants :

- `proton`— Permet aux contributeurs d'accéder en lecture seule aux API. AWS Proton

Détails de l'autorisation

Cette politique inclut les autorisations suivantes.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codepipeline:ListPipelineExecutions",
        "codepipeline:ListPipelines",
        "codepipeline:GetPipeline",
        "codepipeline:GetPipelineState",
        "codepipeline:GetPipelineExecution",
        "proton:GetAccountRoles",

```



```
"proton:GetAccountSettings",
"proton:GetEnvironment",
"proton:GetEnvironmentAccountConnection",
"proton:GetEnvironmentTemplate",
"proton:GetEnvironmentTemplateMajorVersion",
"proton:GetEnvironmentTemplateMinorVersion",
"proton:GetEnvironmentTemplateVersion",
"proton:GetRepository",
"proton:GetRepositorySyncStatus",
"proton:GetResourcesSummary",
"proton:GetService",
"proton:GetServiceInstance",
"proton:GetServiceTemplate",
"proton:GetServiceTemplateMajorVersion",
"proton:GetServiceTemplateMinorVersion",
"proton:GetServiceTemplateVersion",
"proton:GetTemplateSyncConfig",
"proton:GetTemplateSyncStatus",
"proton:ListEnvironmentAccountConnections",
"proton:ListEnvironmentOutputs",
"proton:ListEnvironmentProvisionedResources",
"proton:ListEnvironments",
"proton:ListEnvironmentTemplateMajorVersions",
"proton:ListEnvironmentTemplateMinorVersions",
"proton:ListEnvironmentTemplates",
"proton:ListEnvironmentTemplateVersions",
"proton:ListRepositories",
"proton:ListRepositorySyncDefinitions",
"proton:ListServiceInstanceOutputs",
"proton:ListServiceInstanceProvisionedResources",
"proton:ListServiceInstances",
"proton:ListServicePipelineOutputs",
"proton:ListServicePipelineProvisionedResources",
"proton:ListServices",
"proton:ListServiceTemplateMajorVersions",
"proton:ListServiceTemplateMinorVersions",
"proton:ListServiceTemplates",
"proton:ListServiceTemplateVersions",
"proton:ListTagsForResource"
],
"Resource": "*"
}
]
```

```
}
```

Politique gérée par AWS : `AWSProtonSyncServiceRolePolicy`

AWS Proton associe cette politique au rôle `AWSServiceRoleForProtonSync` lié au service qui permet d'effectuer la AWS Proton synchronisation des modèles.

Cette politique accorde des autorisations qui permettent un accès limité aux AWS Proton actions et aux autres actions de AWS service qui en AWS Proton dépendent.

La politique inclut les espaces de noms d'actions clés suivants :

- `proton`— Permet de AWS Proton synchroniser un accès limité aux AWS Proton API.
- `codestar-connections`— Permet de AWS Proton synchroniser un accès limité aux CodeConnections API.

Pour plus d'informations sur les détails des autorisations pour le `AWSProtonSyncServiceRolePolicy`, consultez la section [Autorisations de rôle liées au service](#) pour AWS Proton

AWS politique gérée : `AWSProtonCodeBuildProvisioningBasicAccess`

Les autorisations CodeBuild doivent exécuter une version pour le AWS Proton CodeBuild provisionnement. Vous pouvez le rattacher `AWSProtonCodeBuildProvisioningBasicAccess` à votre rôle CodeBuild d'approvisionnement.

Cette politique accorde les autorisations minimales nécessaires au fonctionnement d'AWS Proton CodeBuild Provisioning. Il accorde des autorisations qui permettent CodeBuild de générer des journaux de construction. Il autorise également Proton à mettre les sorties Infrastructure as Code (IaC) à la disposition des AWS Proton utilisateurs. Il ne fournit pas les autorisations nécessaires aux outils IaC pour gérer l'infrastructure.

La politique inclut les espaces de noms d'actions clés suivants :

- `logs`- Permet CodeBuild de générer des journaux de construction. Sans cette autorisation, il ne CodeBuild pourra pas démarrer.
- `proton`- Permet à une commande de CodeBuild provisionnement d'appeler `aws proton notify-resource-deployment-status-change` pour mettre à jour les sorties IaC pour une ressource donnée AWS Proton.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream",
        "logs:CreateLogGroup",
        "logs:PutLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:/aws/codebuild/AWSProton-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "proton:NotifyResourceDeploymentStatusChange",
      "Resource": "arn:aws:proton:*:*:*"
    }
  ]
}
```

AWSpolitique gérée : AWSProtonCodeBuildProvisioningServiceRolePolicy

AWS Protonassocie cette politique au rôle AWSServiceRoleForProtonCodeBuildProvisioning lié au service qui permet d'AWS Protoneffectuer un provisionnement CodeBuild basé.

Cette politique accorde des autorisations qui permettent un accès limité aux actions de AWS service qui AWS Proton dépendent de.

La politique inclut les espaces de noms d'actions clés suivants :

- `cloudformation`— Autorise le provisionnement AWS Proton CodeBuild basé sur un accès limité aux AWS CloudFormation API.
- `codebuild`— Autorise le provisionnement AWS Proton CodeBuild basé sur un accès limité aux CodeBuild API.

- `iam`— Permet aux administrateurs de transmettre des rôles à AWS Proton. Cela est nécessaire pour AWS Proton pouvoir effectuer des appels d'API vers d'autres services au nom de l'administrateur.
- `servicequotas`— Permet de AWS Proton vérifier la limite de compilation CodeBuild simultanée, ce qui garantit une mise en file d'attente correcte des builds.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:CreateStack",
        "cloudformation:CreateChangeSet",
        "cloudformation>DeleteChangeSet",
        "cloudformation>DeleteStack",
        "cloudformation:UpdateStack",
        "cloudformation:DescribeStacks",
        "cloudformation:DescribeStackEvents",
        "cloudformation:ListStackResources"
      ],
      "Resource": [
        "arn:aws:cloudformation:*:*:stack/AWSProton-CodeBuild-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:CreateProject",
        "codebuild>DeleteProject",
        "codebuild:UpdateProject",
        "codebuild:StartBuild",
        "codebuild:StopBuild",
        "codebuild:RetryBuild",
        "codebuild:BatchGetBuilds",
        "codebuild:BatchGetProjects"
      ],
      "Resource": "arn:aws:codebuild:*:*:project/AWSProton*"
    }
  ]
}
```

```
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "*",
      "Condition": {
        "StringEqualsIfExists": {
          "iam:PassedToService": "codebuild.amazonaws.com"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "servicequotas:GetServiceQuota"
      ],
      "Resource": "*"
    }
  ]
}
```

AWS stratégie gérée : AwsProtonServiceGitSyncServiceRole Politique

AWS Proton associe cette politique au rôle lié au service AwsProtonServiceGitSyncServiceRole Policy qui permet d'AWS Proton effectuer la synchronisation des services.

Cette politique accorde des autorisations qui permettent un accès limité aux AWS Proton actions et aux autres actions de AWS service qui en AWS Proton dépendent.

La politique inclut les espaces de noms d'actions clés suivants :

- `proton`— Permet à AWS Proton sync d'accéder de manière limitée aux API AWS Proton.

Pour plus d'informations sur les détails des autorisations pour la AwsProtonServiceGitSyncServiceRole politique, consultez la section [Autorisations de rôle liées au service](#) pour. AWS Proton

Mises à jour AWS Proton vers des politiques gérées par AWS

Consultez le détail des mises à jour des politiques gérées par AWS pour AWS Proton depuis que ce service a commencé à suivre ces modifications. Pour obtenir des alertes automatiques concernant

les modifications apportées à cette page, abonnez-vous au flux RSS sur la page d'historique du document AWS Proton.

Modification	Description	Date
AWSProtonCodeBuildProvisioningServiceRolePolicy - mise à jour d'une politique existante	AWS Protona mis à jour cette politique pour ajouter des autorisations afin de garantir que les comptes disposent de la limite de création CodeBuild simultanée nécessaire pour utiliser CodeBuild Provisioning.	12 mai 2023
AwsProtonServiceGitSyncServiceRolePolitique — Nouvelle politique	AWS Protona ajouté une nouvelle politique permettant d'AWS Proton effectuer la synchronisation des services. La politique est utilisée dans le rôle AWSServiceRoleForProtonServiceSync lié au service.	31 mars 2023
AWSProtonDeveloperAccess – Mise à jour d'une politique existante	AWS Protona ajouté une nouvelle <code>GetResourcesSummary</code> action qui vous permet d'afficher un résumé de vos modèles, des ressources de modèles déployées et des ressources obsolètes.	18 novembre 2022
AWSProtonReadOnlyAccess – Mise à jour d'une politique existante	AWS Protona ajouté une nouvelle <code>GetResourcesSummary</code> action qui vous permet d'afficher un résumé de vos modèles, des ressources de modèles	18 novembre 2022

Modification	Description	Date
	déployées et des ressources obsolètes.	
AWSProtonCodeBuildProvisioningBasicAccess – Nouvelle politique	AWS Protona ajouté une nouvelle politique qui lui donne CodeBuild les autorisations nécessaires pour exécuter une version pour le AWS Proton CodeBuild provisionnement.	16 novembre 2022
AWSProtonCodeBuildProvisioningServiceRolePolicy – Nouvelle politique	AWS Protona ajouté une nouvelle politique permettant d'AWS Proton effectuer des opérations liées au provisionnement CodeBuild basé. La politique est utilisée dans le rôle AWSServiceRoleForProtonCodeBuildProvisioning lié au service.	2 septembre 2022
AWSProtonFullAccess – Mise à jour d'une politique existante	AWS Protona mis à jour cette politique afin de permettre l'accès aux nouvelles opérations d'AWS ProtonAPI et de résoudre les problèmes d'autorisation pour certaines opérations de AWS Proton console.	30 mars 2022

Modification	Description	Date
AWSProtonDeveloperAccess – Mise à jour d'une politique existante	AWS Proton mettre à jour cette politique afin de fournir l'accès aux nouvelles opérations d'AWS ProtonAPI et de résoudre les problèmes d'autorisation pour certaines opérations de AWS Proton console.	30 mars 2022
AWSProtonReadOnlyAccess – Mise à jour d'une politique existante	AWS Proton mettre à jour cette politique afin de fournir l'accès aux nouvelles opérations d'AWS ProtonAPI et de résoudre les problèmes d'autorisation pour certaines opérations de AWS Proton console.	30 mars 2022
AWSProtonSyncServiceRolePolicy – Nouvelle politique	AWS Proton a ajouté une nouvelle politique permettant d'AWS Proton effectuer des opérations liées à la synchronisation des modèles. La politique est utilisée dans le rôle AWSServiceRoleForProtonSync lié au service.	23 novembre 2021
AWSProtonFullAccess – Nouvelle politique	AWS Proton a ajouté une nouvelle politique pour fournir un accès aux rôles administratifs aux opérations de l'AWS ProtonAPI et à la AWS Proton console.	9 juin 2021

Modification	Description	Date
AWSProtonDeveloperAccess – Nouvelle politique	AWS Protona ajouté une nouvelle politique pour fournir un accès aux rôles de développeur aux opérations de l'AWS ProtonAPI et à la AWS Proton console.	9 juin 2021
AWSProtonReadOnlyAccess – Nouvelle politique	AWS Protona ajouté une nouvelle politique pour fournir un accès en lecture seule aux opérations de AWS Proton l'API et à la AWS Proton console.	9 juin 2021
AWS Proton a démarré le suivi des modifications.	AWS Proton a commencé à suivre les modifications pour ses politiques gérées par AWS.	9 juin 2021

Utilisation des rôles liés aux services pour AWS Proton

AWS Proton utilise des rôles AWS Identity and Access Management (IAM) [liés à un service](#). Un rôle lié à un service est un type unique de rôle IAM lié directement à AWS Proton. Les rôles liés à un service sont prédéfinis par AWS Proton et comprennent toutes les autorisations nécessaires au service pour appeler d'autres services AWS en votre nom.

Rubriques

- [Utilisation des rôles pour la AWS Proton synchronisation](#)
- [Utilisation des rôles pour le provisionnement CodeBuild basé](#)

Utilisation des rôles pour la AWS Proton synchronisation

AWS Proton utilise des rôles AWS Identity and Access Management (IAM) [liés à un service](#). Un rôle lié à un service est un type unique de rôle IAM lié directement à AWS Proton. Les rôles liés à

un service sont prédéfinis par AWS Proton et comprennent toutes les autorisations nécessaires au service pour appeler d'autres services AWS en votre nom.

Un rôle lié à un service permet d'utiliser AWS Proton plus facilement, car vous n'avez pas besoin d'ajouter manuellement les autorisations requises. AWS Proton définit les autorisations de ses rôles liés à un service et, sauf définition contraire, seul AWS Proton peut endosser ses rôles. Les autorisations définies comprennent la politique d'approbation et la politique d'autorisation. De plus, cette politique d'autorisation ne peut pas être attachée à une autre entité IAM.

Vous pouvez supprimer un rôle lié à un service uniquement après la suppression préalable de ses ressources connexes. Vos ressources AWS Proton sont ainsi protégées, car vous ne pouvez pas involontairement supprimer l'autorisation d'accéder aux ressources.

Pour plus d'informations sur les autres services prenant en charge les rôles liés à un service, consultez les [AWS services opérationnels avec IAM](#) et recherchez les services présentant la mention Yes (Oui) dans la colonne Service-linked roles (Rôles liés à un service). Sélectionnez un Oui ayant un lien pour consulter la documentation du rôle lié à un service, pour ce service.

Autorisations des rôles liés à un service pour AWS Proton

AWS Proton utilise deux rôles liés à un service nommés `AWSServiceRoleForProtonSync` et `AWSServiceRoleForProtonServiceSync`.

Le rôle `AWSServiceRoleForProtonSync` lié à un service fait confiance aux services suivants pour assumer le rôle :

- `sync.proton.amazonaws.com`

La politique d'autorisations de rôle nommée `AWSProtonSyncServiceRolePolicy` AWS Proton permet d'effectuer les actions suivantes sur les ressources spécifiées :

- Action : créer, gérer et lire des AWS Proton modèles et des versions de modèles
- Action : utiliser la connexion activée `CodeConnections`

`AWSProtonSyncServiceRolePolicy`

Cette politique inclut les autorisations suivantes :

```
{
```

```

"Version": "2012-10-17",
"Statement": [
  {
    "Sid": "SyncToProton",
    "Effect": "Allow",
    "Action": [
      "proton:UpdateServiceTemplateVersion",
      "proton:UpdateServiceTemplate",
      "proton:UpdateEnvironmentTemplateVersion",
      "proton:UpdateEnvironmentTemplate",
      "proton:GetServiceTemplateVersion",
      "proton:GetServiceTemplate",
      "proton:GetEnvironmentTemplateVersion",
      "proton:GetEnvironmentTemplate",
      "proton>DeleteServiceTemplateVersion",
      "proton>DeleteEnvironmentTemplateVersion",
      "proton>CreateServiceTemplateVersion",
      "proton>CreateServiceTemplate",
      "proton>CreateEnvironmentTemplateVersion",
      "proton>CreateEnvironmentTemplate",
      "proton:ListEnvironmentTemplateVersions",
      "proton:ListServiceTemplateVersions",
      "proton>CreateEnvironmentTemplateMajorVersion",
      "proton>CreateServiceTemplateMajorVersion"
    ],
    "Resource": "*"
  },
  {
    "Sid": "AccessGitRepos",
    "Effect": "Allow",
    "Action": [
      "codestar-connections:UseConnection"
    ],
    "Resource": "arn:aws:codestar-connections:*:*:connection/*"
  }
]
}

```

Pour plus d'informations sur le `AWSProtonSyncServiceRolePolicy`, voir la [politique AWS gérée : `AWSProtonSyncServiceRolePolicy`](#).

Le rôle `AWSServiceRoleForProtonServiceSync` lié à un service fait confiance aux services suivants pour assumer le rôle :

- `service-sync.proton.amazonaws.com`

La politique d'autorisations de rôle nommée `AWSServiceRoleForProtonServiceSync` AWS Proton permet d'effectuer les actions suivantes sur les ressources spécifiées :

- Action : créer, gérer et lire des AWS Proton services et des instances de service

`AwsProtonServiceGitSyncServiceRolePolitique`

Cette politique inclut les autorisations suivantes :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ProtonServiceSync",
      "Effect": "Allow",
      "Action": [
        "proton:GetService",
        "proton:UpdateService",
        "proton:UpdateServicePipeline",
        "proton:CreateServiceInstance",
        "proton:GetServiceInstance",
        "proton:UpdateServiceInstance",
        "proton:ListServiceInstances",
        "proton:GetComponent",
        "proton:CreateComponent",
        "proton:ListComponents",
        "proton:UpdateComponent",
        "proton:GetEnvironment",
        "proton:CreateEnvironment",
        "proton:ListEnvironments",
        "proton:UpdateEnvironment"
      ],
      "Resource": "*"
    }
  ]
}
```

Pour plus d'informations sur le `AwsProtonServiceSyncServiceRolePolicy`, voir la [politique AWS gérée : `AwsProtonServiceSyncServiceRolePolicy`](#).

Vous devez configurer les autorisations de manière à permettre à une entité IAM (comme un utilisateur, un groupe ou un rôle) de créer, modifier ou supprimer un rôle lié à un service. Pour plus d'informations, consultez [Autorisations de rôles liés à un service](#) dans le Guide de l'utilisateur IAM.

Création d'un rôle lié à un service pour AWS Proton

Vous n'avez pas besoin de créer manuellement un rôle lié à un service. Lorsque vous configurez un référentiel ou un service à synchroniser AWS Proton dans le AWS Management Console, le AWS CLI ou l'AWSAPI, vous AWS Proton créez le rôle lié au service pour vous.

Si vous supprimez ce rôle lié à un service et que vous avez ensuite besoin de le recréer, vous pouvez utiliser la même procédure pour recréer le rôle dans votre compte. Lorsque vous configurez un référentiel ou un service pour la synchronisation AWS Proton, AWS Proton crée à nouveau le rôle lié au service pour vous.

Pour recréer le rôle `AWSServiceRoleForProtonSync` lié à un service, vous devez configurer un référentiel pour la synchronisation, et pour le recréer `AWSServiceRoleForProtonServiceSync`, vous devez configurer un service pour la synchronisation.

Modification d'un rôle lié à un service pour AWS Proton

AWS Protonne vous permet pas de modifier le rôle `AWSServiceRoleForProtonSync` lié au service. Une fois que vous avez créé un rôle lié à un service, vous ne pouvez pas changer le nom du rôle, car plusieurs entités peuvent faire référence au rôle. Néanmoins, vous pouvez modifier la description du rôle à l'aide d'IAM. Pour en savoir plus, consultez [Modification d'un rôle lié à un service](#) dans le guide de l'utilisateur IAM.

Suppression d'un rôle lié à un service pour AWS Proton

Il n'est pas nécessaire de supprimer le `AWSServiceRoleForProtonSync` rôle manuellement. Lorsque vous supprimez tous les référentiels AWS Proton liés pour les synchroniser dans l'APIAWS Management Console, le AWS CLI ou l'AWSAPI, vous AWS Proton nettoyez les ressources et supprimez le rôle lié au service pour vous.

Régions prises en charge pour les rôles liés à un service AWS Proton

AWS Proton prend en charge l'utilisation des rôles liés à un service dans toutes les Régions AWS où le service est disponible. Pour plus d'informations, consultez [Points de terminaison et quotas AWS Proton](#) dans le document Références générales AWS.

Utilisation des rôles pour le provisionnement CodeBuild basé

AWS Proton utilise des rôles AWS Identity and Access Management (IAM) [liés à un service](#). Un rôle lié à un service est un type unique de rôle IAM lié directement à AWS Proton. Les rôles liés à un service sont prédéfinis par AWS Proton et comprennent toutes les autorisations nécessaires au service pour appeler d'autres services AWS en votre nom.

Un rôle lié à un service permet d'utiliser AWS Proton plus facilement, car vous n'avez pas besoin d'ajouter manuellement les autorisations requises. AWS Proton définit les autorisations de ses rôles liés à un service et, sauf définition contraire, seul AWS Proton peut endosser ses rôles. Les autorisations définies comprennent la politique d'approbation et la politique d'autorisation. De plus, cette politique d'autorisation ne peut pas être attachée à une autre entité IAM.

Vous pouvez supprimer un rôle lié à un service uniquement après la suppression préalable de ses ressources connexes. Vos ressources AWS Proton sont ainsi protégées, car vous ne pouvez pas involontairement supprimer l'autorisation d'accéder aux ressources.

Pour plus d'informations sur les autres services prenant en charge les rôles liés à un service, consultez les [AWS services opérationnels avec IAM](#) et recherchez les services présentant la mention Yes (Oui) dans la colonne Service-linked roles (Rôles liés à un service). Sélectionnez un Oui ayant un lien pour consulter la documentation du rôle lié à un service, pour ce service.

Autorisations des rôles liés à un service pour AWS Proton

AWS Proton utilise le rôle lié à un service nommé `AWSServiceRoleForProtonCodeBuildProvisioning`— Un rôle lié à un service pour AWS Proton CodeBuild le provisionnement.

Le rôle `AWSServiceRoleForProtonCodeBuildProvisioning` lié à un service fait confiance aux services suivants pour assumer le rôle :

- `codebuild.proton.amazonaws.com`

La politique d'autorisations de rôle nommée `AWSProtonCodeBuildProvisioningServiceRolePolicy` AWS Proton permet d'effectuer les actions suivantes sur les ressources spécifiées :

- Action : créer, gérer et lire sur les AWS CloudFormation piles et les transformations
- Action : créer, gérer et lire des CodeBuild projets et des builds

AWSProtonCodeBuildProvisioningServiceRolePolicy

Cette politique inclut les autorisations suivantes :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:CreateStack",
        "cloudformation:CreateChangeSet",
        "cloudformation>DeleteChangeSet",
        "cloudformation>DeleteStack",
        "cloudformation:UpdateStack",
        "cloudformation:DescribeStacks",
        "cloudformation:DescribeStackEvents",
        "cloudformation:ListStackResources"
      ],
      "Resource": [
        "arn:aws:cloudformation:*:*:stack/AWSProton-CodeBuild-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:CreateProject",
        "codebuild>DeleteProject",
        "codebuild:UpdateProject",
        "codebuild:StartBuild",
        "codebuild:StopBuild",
        "codebuild:RetryBuild",
        "codebuild:BatchGetBuilds",
        "codebuild:BatchGetProjects"
      ],
      "Resource": "arn:aws:codebuild:*:*:project/AWSProton*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "*",
      "Condition": {
        "StringEqualsIfExists": {
          "iam:PassedToService": "codebuild.amazonaws.com"
        }
      }
    }
  ]
}
```

```
    }
  }
},
{
  "Effect": "Allow",
  "Action": [
    "servicequotas:GetServiceQuota"
  ],
  "Resource": "*"
}
]
```

Vous devez configurer les autorisations de manière à permettre à une entité IAM (comme un utilisateur, un groupe ou un rôle) de créer, modifier ou supprimer un rôle lié à un service. Pour plus d'informations, consultez [Autorisations de rôles liés à un service](#) dans le Guide de l'utilisateur IAM.

Création d'un rôle lié à un service pour AWS Proton

Vous n'avez pas besoin de créer manuellement un rôle lié à un service. Lorsque vous créez un environnement qui utilise le provisionnement CodeBuild basé AWS Proton dans l'APIAWS Management Console, le ou l'AWSAPIAWS CLI, vous AWS Proton créez le rôle lié au service pour vous.

Si vous supprimez ce rôle lié à un service et que vous avez ensuite besoin de le recréer, vous pouvez utiliser la même procédure pour recréer le rôle dans votre compte. Lorsque vous créez un environnement qui utilise le provisionnement CodeBuild basé sur le provisionnement dansAWS Proton, il AWS Proton crée à nouveau le rôle lié au service pour vous.

Modification d'un rôle lié à un service pour AWS Proton

AWS Protonne vous permet pas de modifier le rôle AWSServiceRoleForProtonCodeBuildProvisioning lié au service. Une fois que vous avez créé un rôle lié à un service, vous ne pouvez pas changer le nom du rôle, car plusieurs entités peuvent faire référence à ce rôle. Néanmoins, vous pouvez modifier la description du rôle à l'aide d'IAM. Pour en savoir plus, consultez [Modification d'un rôle lié à un service](#) dans le guide de l'utilisateur IAM.

Suppression d'un rôle lié à un service pour AWS Proton

Si vous n'avez plus besoin d'utiliser une fonction ou un service qui nécessite un rôle lié à un service, nous vous recommandons de supprimer ce rôle. De cette façon, vous n'avez aucune entité

inutilisée qui n'est pas surveillée ou gérée activement. Toutefois, vous devez supprimer tous les environnements et services (instances et pipelines) qui utilisent le provisionnement CodeBuild basé AWS Proton avant de pouvoir le supprimer manuellement.

Suppression manuelle du rôle lié au service

Utilisez la console IAM, l'AWS CLI ou l'API AWS pour supprimer le rôle lié à un service `AWSServiceRoleForProtonCodeBuildProvisioning`. Pour plus d'informations, consultez [Suppression d'un rôle lié à un service](#) dans le Guide de l'utilisateur IAM.

Régions prises en charge pour les rôles liés à un service AWS Proton

AWS Proton prend en charge l'utilisation des rôles liés à un service dans toutes les Régions AWS où le service est disponible. Pour plus d'informations, consultez [Points de terminaison et quotas AWS Proton](#) dans le document Références générales AWS.

Résolution des problèmes d'identité et d'accès avec AWS Proton

Utilisez les informations suivantes pour identifier et résoudre les problèmes courants que vous pouvez rencontrer lorsque vous travaillez avec AWS Proton et IAM.

Rubriques

- [Je ne suis pas autorisé à effectuer une action dans AWS Proton](#)
- [Je ne suis pas autorisé à effectuer iam : PassRole](#)
- [Je veux autoriser des personnes extérieures à mon Compte AWS à accéder à mes ressources AWS Proton](#)

Je ne suis pas autorisé à effectuer une action dans AWS Proton

Si la AWS Management Console indique que vous n'êtes pas autorisé à exécuter une action, vous devez contacter votre administrateur pour obtenir de l'aide. Votre administrateur est la personne qui vous a fourni vos informations de connexion.

L'exemple d'erreur suivant se produit quand l'utilisateur IAM `mateojackson` tente d'utiliser la console pour afficher des informations détaillées sur une ressource `my-example-widget` fictive, mais ne dispose pas des autorisations `proton:GetWidget` fictives.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
proton:GetWidget on resource: my-example-widget
```

Dans ce cas, Mateo demande à son administrateur de mettre à jour ses politiques pour lui permettre d'accéder à la ressource *my-example-widget* à l'aide de l'action proton : *GetWidget*.

Je ne suis pas autorisé à effectuer iam : PassRole

Si vous recevez une erreur selon laquelle vous n'êtes pas autorisé à exécuter `iam:PassRole` l'action, vos stratégies doivent être mises à jour afin de vous permettre de transmettre un rôle à AWS Proton.

Certains Services AWS vous permettent de transmettre un rôle existant à ce service, au lieu de créer une nouvelle fonction de service ou un rôle lié à un service. Pour ce faire, un utilisateur doit disposer des autorisations nécessaires pour transmettre le rôle au service.

L'exemple d'erreur suivant se produit lorsqu'un utilisateur IAM nommé `marymajor` essaie d'utiliser la console pour exécuter une action dans AWS Proton. Toutefois, l'action nécessite que le service ait des autorisations accordées par une fonction de service. Mary ne dispose pas des autorisations nécessaires pour transférer le rôle au service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

Dans ce cas, les politiques de Mary doivent être mises à jour pour lui permettre d'exécuter l'action `iam:PassRole`.

Si vous avez encore besoin d'aide, contactez votre administrateur AWS. Votre administrateur vous a fourni vos informations de connexion.

Je veux autoriser des personnes extérieures à mon Compte AWS à accéder à mes ressources AWS Proton

Vous pouvez créer un rôle que les utilisateurs provenant d'autres comptes ou les personnes extérieures à votre organisation pourront utiliser pour accéder à vos ressources. Vous pouvez spécifier qui est autorisé à assumer le rôle. Pour les services qui prennent en charge les politiques basées sur les ressources ou les listes de contrôle d'accès (ACL), vous pouvez utiliser ces politiques pour donner l'accès à vos ressources.

Pour en savoir plus, consultez les éléments suivants :

- Pour savoir si AWS Proton prend en charge ces fonctionnalités, consultez [Fonctionnement de AWS Proton avec IAM](#).

- Pour savoir comment octroyer l'accès à vos ressources à des Comptes AWS dont vous êtes propriétaire, consultez [Fournir l'accès à un utilisateur IAM dans un autre Compte AWS que vous possédez](#) dans le Guide de l'utilisateur IAM.
- Pour savoir comment octroyer l'accès à vos ressources à des tiers Comptes AWS, consultez [Fournir l'accès aux Comptes AWS appartenant à des tiers](#) dans le Guide de l'utilisateur IAM.
- Pour savoir comment fournir un accès par le biais de la fédération d'identité, consultez [Fournir un accès à des utilisateurs authentifiés en externe \(fédération d'identité\)](#) dans le Guide de l'utilisateur IAM.
- Pour découvrir quelle est la différence entre l'utilisation des rôles et l'utilisation des politiques basées sur les ressources pour l'accès entre comptes, consultez [Différence entre les rôles IAM et les politiques basées sur les ressources](#) dans le Guide de l'utilisateur IAM.

Configuration et analyse des vulnérabilités dans AWS Proton

AWS Protonne fournit pas de correctifs ou de mises à jour pour le code fourni par le client. Les clients sont responsables de la mise à jour et de l'application des correctifs à leur propre code, y compris le code source de leurs services et applications exécutés sur AWS Protonet le code fourni dans leurs offres groupées de modèles de service et d'environnement.

Les clients sont responsables de la mise à jour et de la correction des ressources de l'infrastructure dans leurs environnements et services. AWS Protonne mettra pas automatiquement à jour ou ne corrigera aucune ressource. Les clients doivent consulter la documentation relative aux ressources de leur architecture afin de comprendre leurs politiques de correctifs respectives.

Outre la fourniture de mises à jour d'environnement et de service demandées par le client à des versions mineures des modèles de service et d'environnement, AWS Protonne fournit pas de correctifs ou de mises à jour des ressources que les clients définissent dans leurs modèles de service et d'environnement et leurs offres groupées de modèles.

Pour plus de détails, consultez les ressources suivantes :

- [Modèle de responsabilité partagée](#)
- [Amazon Web Services : Présentation des processus de sécurité](#)

Protection des données dans AWS Proton

AWS Proton est conforme au modèle de [responsabilité AWS partagée modèle](#) de de qui inclut les réglementations et directives relatives à la protection des données. AWS est responsable de la protection de l'infrastructure mondiale qui fait fonctionner tous les Services AWS. AWS gère le contrôle sur des données hébergées sur cette infrastructure, y compris les contrôles de configuration de la sécurité afin de gérer les données personnelles et les contenus des clients. AWS les clients et les partenaires APN, agissant en tant que contrôleurs de données ou processeurs de données, sont responsables de toutes les données personnelles qu'ils mettent dans AWS Cloud

Pour des raisons de protection des données, nous vous recommandons de protéger votre compte AWS vos informations d'identification et de configurer des comptes utilisateurs individuels avec AWS Identity and Access Management (IAM), afin que chaque utilisateur reçoive uniquement les autorisations nécessaires pour accomplir ses tâches. Nous vous recommandons également de sécuriser vos données comme indiqué ci-dessous :

- Utilisez l'authentification multifactorielle (MFA) avec chaque compte.
- Utilisez les certificats SSL/TLS pour communiquer avec les ressources AWS. Nous vous recommandons le certificats TLS 1.2 ou une version ultérieure.
- Configurez une API (Interface de programmation) et le journal de l'activité des utilisateurs avec AWS CloudTrail.
- Utilisez des solutions de chiffrement AWS, ainsi que tous les contrôles de sécurité par défaut au sein des Services AWS.

Nous vous recommandons vivement de ne jamais placer d'informations identifiables sensibles, telles que les numéros de compte de vos clients, dans des champs de texte de formulaire comme Name (Nom). Cela inclut lorsque vous utilisez AWS Proton ou Services AWS utilisez la console AWS CLI, l'API ou AWS les SDK. Toutes les données que vous entrez dans des champs de texte de formulaire pour identifiables de ressources ou des éléments similaires liés à la gestion des AWS ressources peuvent être récupérées pour insertion dans des journaux de diagnostic. Lorsque vous fournissez une URL à un serveur externe, n'incluez pas les informations d'identification non chiffrées dans l'URL pour valider votre demande adressée au serveur.

Pour en savoir plus sur la protection des données, consultez le billet de blog [Modèle de responsabilité partagée AWS et RGPD](#) sur le Blog sur la sécurité d'AWS.

Chiffrement côté serveur au repos

Si vous choisissez de chiffrer les données sensibles de vos ensembles de modèles au repos dans le compartiment S3 dans lequel vous stockez vos ensembles de modèles, vous devez utiliser une clé SSE-S3 ou SSE-KMS pour AWS Proton permettre de récupérer les ensembles de modèles afin qu'ils puissent être joints à un AWS Proton modèle enregistré.

Chiffrement en transit

Toutes les communications de service à service sont chiffrées en transit à l'aide de SSL/TLS.

AWS Proton gestion des clés de chiffrement

Toutes AWS Proton les données des clients y sont cryptées par défaut à l'aide d'une clé AWS Proton propriétaire. Si vous fournissez une AWS KMS clé détenue et gérée par le client, toutes les données du client sont cryptées à l'aide de la clé fournie par le client, comme décrit dans les paragraphes suivants.

Lorsque vous créez un AWS Proton modèle, vous spécifiez votre clé et AWS Proton utilise vos informations d'identification pour créer une autorisation qui permet AWS Proton d'utiliser votre clé.

Si vous retirez manuellement l'autorisation ou si vous désactivez ou supprimez la clé spécifiée, vous AWS Proton ne pourrez pas lire les données cryptées par la clé spécifiée et renvoyées `ValidationException`.

Contexte de chiffrement AWS Proton

AWS Proton prend en charge les en-têtes de contexte de chiffrement. Un contexte de chiffrement est un ensemble facultatif de paires clé-valeur qui peut contenir des informations contextuelles supplémentaires sur les données. Pour des informations générales sur le contexte de chiffrement, consultez la section [Concepts AWS Key Management Service - Contexte de chiffrement](#) du Guide du développeur AWS Key Management Service.

Un contexte de chiffrement est un ensemble de paires valeur clé qui contient des données non secrètes arbitraires. Lorsque vous incluez un contexte de chiffrement dans une demande de chiffrement de données, il lie AWS KMS cryptographiquement le contexte de chiffrement aux données chiffrées. Pour déchiffrer les données, vous devez transmettre le même contexte de chiffrement.

Les clients peuvent utiliser le contexte de chiffrement pour identifier l'utilisation de leur clé gérée par le client dans les enregistrements d'audit et les journaux. Il apparaît également en texte brut dans les journaux, comme AWS CloudTrail et Amazon CloudWatch Logs.

AWS Protonne prend en compte aucun contexte de chiffrement spécifié par le client ou spécifié en externe.

AWS Protonajoute le contexte de chiffrement suivant.

```
{
  "aws:proton:template": "<proton-template-arn>",
  "aws:proton:resource": "<proton-resource-arn>"
}
```

Le premier contexte de chiffrement identifie leAWS Proton modèle auquel la ressource est associée et sert également de contrainte pour les autorisations et les autorisations de clé gérées par le client.

Le second contexte de chiffrement identifie laAWS Proton ressource cryptée.

Les exemples suivants illustrent l'utilisation du contexte deAWS Proton chiffrement.

Développeur créant une instance de service.

```
{
  "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"
}
```

Un administrateur qui crée un modèle.

```
{
  "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"
}
```

Sécurité de l'infrastructure dans AWS Proton

En tant que service géré, AWS Proton il est protégé par la sécurité du réseau AWS mondial. Pour plus d'informations sur les services AWS de sécurité et sur la manière dont AWS l'infrastructure est protégée, consultez la section [Sécurité du AWS cloud](#). Pour concevoir votre AWS environnement en utilisant les meilleures pratiques en matière de sécurité de l'infrastructure, consultez la section [Protection de l'infrastructure](#) dans le cadre AWS bien architecturé du pilier de sécurité.

Vous utilisez des appels d'API AWS publiés pour accéder AWS Proton via le réseau. Les clients doivent prendre en charge les éléments suivants :

- Protocole TLS (Transport Layer Security). Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Ses suites de chiffrement PFS (Perfect Forward Secrecy) comme DHE (Ephemeral Diffie-Hellman) ou ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). La plupart des systèmes modernes tels que Java 7 et les versions ultérieures prennent en charge ces modes.

En outre, les demandes doivent être signées à l'aide d'un ID de clé d'accès et d'une clé d'accès secrète associée à un principal IAM. Vous pouvez également utiliser [AWS Security Token Service](#) (AWS STS) pour générer des informations d'identification de sécurité temporaires et signer les demandes.

Pour améliorer l'isolation du réseau, vous pouvez utiliser AWS PrivateLink les méthodes décrites dans la section suivante.

AWS Proton et points de terminaison VPC d'interface ()AWS PrivateLink

Vous pouvez établir une connexion privée entre votre VPC et créer un point de AWS Proton terminaison VPC d'interface. Les points de terminaison d'interface sont alimentés par [AWS PrivateLink](#) une technologie qui vous permet d'accéder à AWS Proton des API de manière privée sans passerelle Internet, périphérique NAT, connexion VPN ou AWS Direct Connect connexion. Les instances de votre VPC n'ont pas besoin d'adresses IP publiques pour communiquer avec AWS Proton les API. Le trafic entre votre VPC et celui qui AWS Proton ne quitte pas le réseau Amazon.

Chaque point de terminaison d'interface est représenté par une ou plusieurs [interfaces réseau Elastic](#) dans vos sous-réseaux.

Pour de plus amples informations, consultez [Points de terminaison VPC \(AWS PrivateLink\)](#) dans le Guide de l'utilisateur Amazon VPC.

Considérations relatives aux points de AWS Proton terminaison VPC

Avant de configurer un point de terminaison VPC d'interface pour AWS Proton, assurez-vous de consulter les [propriétés et les limites du point de terminaison d'interface](#) dans le guide de l'utilisateur Amazon VPC.

AWS Proton permet d'appeler toutes ses actions d'API depuis votre VPC.

Les politiques de point de terminaison VPC sont prises en charge pour AWS Proton. Par défaut, l'accès complet à AWS Proton est autorisé via le point de terminaison. Pour plus d'informations, consultez [Contrôle de l'accès aux services avec points de terminaison d'un VPC](#) dans le Guide de l'utilisateur Amazon VPC.

Création d'un point de terminaison VPC d'interface pour AWS Proton

Vous pouvez créer un point de terminaison VPC pour le AWS Proton service à l'aide de la console Amazon VPC ou du `awscli`. AWS Command Line Interface AWS CLI Pour plus d'informations, consultez [Création d'un point de terminaison d'interface](#) dans le Guide de l'utilisateur Amazon VPC.

Créez un point de terminaison VPC à l'AWS Proton aide du nom de service suivant :

- `com.amazonaws.region.proton`

Si vous activez le DNS privé pour le point de terminaison, vous pouvez envoyer des demandes d'API AWS Proton en utilisant son nom DNS par défaut pour la région, par exemple, `proton.region.amazonaws.com`.

Pour plus d'informations, consultez [Accès à un service via un point de terminaison d'interface](#) dans le Guide de l'utilisateur Amazon VPC.

Création d'une politique de point de terminaison VPC pour AWS Proton

Vous pouvez attacher une stratégie de point de terminaison à votre point de terminaison d'un VPC qui contrôle l'accès à AWS Proton. La politique spécifie les informations suivantes :

- Le principal qui peut exécuter des actions.
- Les actions qui peuvent être effectuées.
- Les ressources sur lesquelles les actions peuvent être exécutées.

Pour plus d'informations, consultez [Contrôle de l'accès aux services avec points de terminaison d'un VPC](#) dans le Guide de l'utilisateur Amazon VPC.

Exemple : politique de point de terminaison VPC pour les actions AWS Proton

Voici un exemple de politique de point de terminaison pour AWS Proton. Lorsqu'elle est attachée à un point de terminaison, cette politique accorde l'accès aux AWS Proton actions répertoriées pour tous les principaux sur toutes les ressources.

```
{
  "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": "*"
    }
  ]
}
```

Journalisation et surveillance dans AWS Proton

La surveillance est un enjeu important pour assurer la fiabilité, la disponibilité et les performances de AWS Proton et vos autres AWS solutions. AWS fournit les outils de surveillance suivants pour surveiller vos instances s'exécutant dans AWS Proton, signaler les problèmes et déclencher des actions automatiques, si nécessaire.

À l'heure actuelle, AWS Proton n'est pas intégré à Amazon CloudWatch Logs ou AWS Trusted Advisor. Les administrateurs peuvent configurer et utiliser CloudWatch pour surveiller d'autres Services AWS tels que définis dans leurs modèles de service et d'environnement. AWS Proton est intégré à AWS CloudTrail.

- Amazon CloudWatch contrôle vos ressources AWS et les applications que vous exécutez sur AWS en temps réel. Vous pouvez collecter et suivre les métriques, créer des tableaux de bord personnalisés, et définir des alarmes qui vous informent ou prennent des mesures lorsqu'une métrique spécifique atteint un seuil que vous spécifiez. Par exemple, vous pouvez faire en sorte que CloudWatch assure le suivi de l'utilisation du processeur ou d'autres métriques de vos instances Amazon EC2 et démarre automatiquement de nouvelles instances lorsque cela est nécessaire. Pour de plus amples informations, veuillez consulter le [Guide de l'utilisateur Amazon CloudWatch](#).
- Amazon CloudWatch Logs vous permet de surveiller, stocker et accéder à vos fichiers journaux à partir d'instances Amazon EC2, de CloudTrail et d'autres sources. CloudWatch Logs peut contrôler les informations contenues dans les fichiers journaux et vous avertir lorsque certains seuils sont atteints. Vous pouvez également archiver vos données de journaux dans une solution de stockage hautement durable. Pour plus d'informations, consultez le [Guide de l'utilisateur Amazon CloudWatch Logs](#).
- AWS CloudTrail capture les appels d'API et les événements associés créés par votre Compte AWS ou au nom de celui-ci et livre les fichiers journaux dans un compartiment Amazon S3 que vous spécifiez. Vous pouvez identifier les utilisateurs et les comptes qui ont appelé AWS, l'adresse IP source à partir de laquelle les appels ont été émis, ainsi que le moment où les appels ont eu lieu. Pour de plus amples informations, veuillez consulter le [Guide de l'utilisateur AWS CloudTrail](#).
- Amazon EventBridge est un service de bus d'événements sans serveur qui facilite la connexion à vos applications avec des données provenant de diverses sources. EventBridge fournit un flux de données en temps réel à partir de vos propres applications, d'applications SaaS (Software-as-a-Service) et Services AWS et achemine ces données vers des cibles telles que Lambda. Cela vous permet de surveiller les événements qui se produisent dans les services et de créer

des architectures basées sur les événements. Pour de plus amples informations, veuillez consulter [Automatisez AWS Proton avec EventBridge](#) et l'[Guide de l'utilisateur EventBridge](#).

Résilience dans AWS Proton

L'infrastructure mondiale d'AWS repose sur les régions AWS et les zones de disponibilité. Les régions AWS fournissent plusieurs zones de disponibilité physiquement séparées et isolées, reliées par un réseau à latence faible, à débit élevé et à forte redondance. Avec les zones de disponibilité, vous pouvez concevoir et exploiter des applications et des bases de données qui basculent automatiquement d'une zone à l'autre sans interruption. Les zones de disponibilité sont plus hautement disponibles, tolérantes aux pannes et évolutives que les infrastructures traditionnelles à un ou plusieurs centres de données.

Pour plus d'informations sur les régions AWS et les zones de disponibilité, consultez [l'infrastructure mondiale d'AWS](#).

Outre l'infrastructure globale d'AWS, AWS Proton propose différentes fonctionnalités qui contribuent à la prise en charge de vos besoins en matière de résilience et de sauvegarde de données.

AWS Proton et sauvegardes

AWS Proton conserve une sauvegarde de toutes les données client. En cas de panne totale, cette sauvegarde peut être utilisée pour restaurer AWS Proton et des données client d'un état valide antérieur.

Bonnes pratiques de sécurité pour AWS Proton

AWS Proton fournit des fonctionnalités de sécurité à prendre en compte lorsque vous développez et mettez en œuvre vos propres stratégies de sécurité. Les bonnes pratiques suivantes doivent être considérées comme des instructions générales et ne représentent pas une solution de sécurité complète. Étant donné que ces bonnes pratiques peuvent ne pas être appropriées ou suffisantes pour votre environnement, considérez-les comme des remarques utiles plutôt que comme des recommandations.

Rubriques

- [Utilisation du contrôle d'accès IAM](#)
- [N'incorporez pas d'informations d'identification dans vos modèles et vos offres groupées de modèles](#)

- [Utiliser le chiffrement pour protéger les données sensibles](#)
- [Utiliser AWS CloudTrail pour afficher et journaliser les appels API](#)

Utilisation du contrôle d'accès IAM

IAM est un service AWS que vous pouvez utiliser pour gérer les utilisateurs et leurs autorisations dans AWS. Vous pouvez utiliser IAM avec AWS Proton pour spécifier les actions que les administrateurs et les développeurs peuvent effectuer, telles que la gestion de modèles, d'environnements ou de services. Vous pouvez utiliser les rôles de service IAM pour autoriser AWS Proton à passer des appels à d'autres services en votre nom.

Pour plus d'informations sur, consultez AWS Proton et les rôles IAM, voir [Identity and Access Management \(Gestion des identités et des accès\) pour AWS Proton](#).

Implémentation d'un accès sur la base des mo Pour de plus amples informations, veuillez consulter [Politiques et autorisations dans IAM](#) dans le AWS Identity and Access Management Guide de l'utilisateur.

N'incorporez pas d'informations d'identification dans vos modèles et vos offres groupées de modèles

Plutôt que d'intégrer des informations sensibles dans votre AWS CloudFormation modèles et ensembles de modèles, nous vous recommandons d'utiliser des références dynamiques dans votre modèle de pile.

Les références dynamiques vous permettent de faire référence de façon concise et efficace à des valeurs externes stockées et gérées dans d'autres services, tels qu'AWS Systems Manager Parameter Store ou AWS Secrets Manager. Lorsque vous utilisez une référence dynamique, CloudFormation récupère la valeur de la référence spécifiée lorsque cela est nécessaire au cours d'opérations de pile et de jeu de modifications et transmet cette valeur à la ressource appropriée. Cependant, CloudFormation ne stocke jamais la valeur de référence réelle. Pour de plus amples informations, veuillez consulter [Utilisation de références dynamiques pour spécifier des valeurs de modèle](#) dans le AWS CloudFormation Guide de l'utilisateur.

[AWS Secrets Manager](#) vous aide à chiffrer, stocker et récupérer en toute sécurité des informations d'identification pour vos bases de données et d'autres services. Le [AWS Systems Manager Parameter Store](#) fournit un stockage sécurisé et hiérarchique des données pour la gestion des données de configuration.

Pour plus d'informations sur la définition des paramètres de modèle, consultez <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/parameters-section-structure.html> dans le AWS CloudFormation Guide de l'utilisateur.

Utiliser le chiffrement pour protéger les données sensibles

WITHIN AWS Proton, toutes les données client sont chiffrées par défaut à l'aide d'un AWS Proton clé possédée.

En tant que membre de l'équipe de la plateforme, vous pouvez fournir une clé gérée par le client pour AWS Proton pour chiffrer et sécuriser vos données sensibles. Cryptez les données sensibles au repos dans votre compartiment S3. Pour plus d'informations, consultez [Protection des données dans AWS Proton](#).

Utiliser AWS CloudTrail pour afficher et journaliser les appels API

AWS CloudTrail effectue le suivi des personnes qui effectuent des appels API dans votre Compte AWS. Les appels d'API sont consignés chaque fois que quelqu'un utilise le AWS Proton API, le AWS Proton console ou AWS Proton AWS CLI commandes. Activez la journalisation et spécifiez un compartiment Amazon S3 pour y stocker les journaux. De cette façon, le cas échéant, vous pourrez déterminer qui a effectué quel AWS Proton appelez votre compte. Pour de plus amples informations, veuillez consulter [Journalisation et surveillance dans AWS Proton](#).

Prévention du député confus entre services

Le problème de député confus est un problème de sécurité dans lequel une entité qui n'est pas autorisée à effectuer une action peut contraindre une entité plus privilégiée à le faire. Dans AWS, l'emprunt d'identité entre services peut entraîner le problème de député confus. L'usurpation d'identité entre services peut se produire lorsqu'un service (le service appelant) appelle un autre service (le service appelé). Le service appelant peut être manipulé pour utiliser ses autorisations afin d'agir sur les ressources d'un autre client de manière à ce qu'il ne soit pas autorisé à y accéder autrement. Pour éviter cela, AWS fournit des outils qui vous aident à protéger vos données pour tous les services avec des principaux de service qui ont eu accès aux ressources de votre compte.

Nous vous recommandons d'utiliser les clés de contexte de condition globale [aws:SourceArn](#) et [aws:SourceAccount](#) dans les politiques de ressources pour limiter les autorisations à la ressource octroyées par AWS Proton à un autre service. Si la valeur `aws:SourceArn` ne contient pas l'ID du compte, tel qu'un ARN de compartiment Amazon S3, vous devez utiliser les deux clés de contexte de condition globale pour limiter les autorisations. Si vous utilisez les deux clés de contexte de condition

globale et que la valeur `aws:SourceArn` contient l'ID de compte, la valeur `aws:SourceAccount` et le compte dans la valeur `aws:SourceArn` doivent utiliser le même ID de compte lorsqu'ils sont utilisés dans la même instruction de politique. Utilisez `aws:SourceArn` si vous souhaitez qu'une seule ressource soit associée à l'accès entre services. Utilisez `aws:SourceAccount` si vous souhaitez autoriser toute ressource de ce compte à être associée à l'utilisation entre services.

Pour `aws:SourceArn` doit être une ressource qui AWS Proton Magasins.

Le moyen le plus efficace de se protéger contre le problème de député confus consiste à utiliser la clé de contexte de condition globale `aws:SourceArn` avec l'ARN complet de la ressource. Si vous ne connaissez pas l'ARN complet de la ressource ou si vous spécifiez plusieurs ressources, utilisez la clé de contexte de condition globale `aws:SourceArn` avec des caractères génériques (*) pour les parties inconnues de l'ARN. Par exemple, `arn:aws::proton*:123456789012:environment/*`.

L'exemple suivant montre comment utiliser `aws:SourceArn` et `aws:SourceAccount` Clés de contexte de condition globales dans AWS Proton pour éviter le problème du député confus.

```
{
  "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 mise en place d'un support Amazon VPC personnalisé

AWS Proton CodeBuild Le provisioning exécute des commandes CLI arbitraires fournies par le client dans un CodeBuild projet situé dans le compte AWS Proton Environment. Ces commandes gèrent

généralement les ressources à l'aide d'un outil IaC (Infrastructure as Code), tel que CDK. Si vous avez des ressources dans un Amazon VPC, vous ne pourrez CodeBuild peut-être pas y accéder. Pour ce faire, CodeBuild prend en charge la possibilité de s'exécuter au sein d'un Amazon VPC spécifique. Voici quelques exemples de cas d'utilisation :

- Récupérez les dépendances à partir de référentiels d'artefacts internes auto-hébergés, tels que PyPI/Maven pour Python, Java et npm/Node.js
- CodeBuild doit accéder à un serveur Jenkins dans un Amazon VPC spécifique pour enregistrer un pipeline.
- Accès aux objets d'un compartiment Amazon S3 configuré pour autoriser l'accès via un point point point point point point point VPC point point point.
- Exécutez des tests d'intégration à partir de votre build sur les données d'une base de données Amazon RDS isolée sur un sous-réseau privé.

Pour plus d'informations, consultez [CodeBuild la documentation VPC](#).

Si vous souhaitez que le CodeBuild provisioning s'exécute dans un VPC personnalisé, il s'agit d'une solution simple. Tout d'abord, vous devez ajouter l'ID VPC, les sous-réseaux et les groupes de sécurité au modèle d'environnement. Vous entrez ensuite ces valeurs dans la spécification d'environnement. Cela se traduira par la création d'un CodeBuild projet pour vous qui cible un VPC donné.

Mise à jour du modèle d'environnement

Schema

L'ID VPC, les sous-réseaux et les groupes de sécurité doivent être ajoutés au schéma du modèle afin qu'ils puissent figurer dans les spécifications d'environnement.

Un exemple `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

```

Cela ajoute trois nouvelles propriétés qui seront utilisées par le manifeste :

- `codebuild_vpc_id`
- `codebuild_subnets`
- `codebuild_security_groups`

Manifeste

Pour configurer les paramètres Amazon VPC dans CodeBuild, une propriété facultative appelée `project_properties` est disponible dans le manifeste du modèle. Le contenu `project_properties` de est ajouté à la AWS CloudFormation pile qui crée le CodeBuild projet. Cela permet d'ajouter non seulement des [AWS CloudFormation propriétés Amazon VPC](#), mais également toute [CodeBuild CloudFormation propriété](#) prise en charge, telle que le délai de génération. Les mêmes données fournies à `proton-inputs.json` sont mises à disposition pour les valeurs de `project_properties`.

Ajoutez cette section à votre `manifest.yaml` :

```

project_properties:
  VpcConfig:
    VpcId: "{{ environment.inputs.codebuild_vpc_id }}"
    Subnets: "{{ environment.inputs.codebuild_subnets }}"
    SecurityGroupIds: "{{ environment.inputs.codebuild_security_groups }}"

```

Voici à quoi `manifest.yaml` peut ressembler le résultat :

```

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 }}"
```

Création de l'environnement

Lorsque vous créez un environnement avec votre modèle compatible CodeBuild Provisioning VPC, vous devez fournir l'identifiant Amazon VPC, les sous-réseaux et les groupes de sécurité.

Pour obtenir la liste de tous les ID Amazon VPC de votre région, exécutez la commande suivante :

```
aws ec2 describe-vpcs
```

Pour obtenir la liste de tous les ID de point point, exécutez :

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=vpc-id"
```

Important

N'incluez que des sous-réseaux privés. CodeBuild échouera si vous fournissez des sous-réseaux publics. Les sous-réseaux publics ont une route par défaut vers une [Internet Gateway](#), ce qui n'est pas le cas des sous-réseaux privés.

Exécutez la commande suivante pour obtenir les ID de point de sécurité. Ces identifiants peuvent également être obtenus via AWS Management Console :

```
aws ec2 describe-security-groups --filters "Name=vpc-id,Values=vpc-id"
```

Les valeurs ressembleront à :

```
vpc-id: vpc-045ch35y28dec3a05
subnets:
  - subnet-04029a82e6ae46968
  - subnet-0f500a9294fc5f26a
security-groups:
  - sg-03bc4c4ce32d67e8d
```

Garantir CodeBuild les autorisations

La prise en charge d'Amazon VPC nécessite certaines autorisations, telles que la possibilité de créer une interface réseau élastique.

Si l'environnement est créé dans la console, entrez ces valeurs lors de l'assistant de création d'environnement. Si vous souhaitez créer l'environnement par programmation, vous aurez l'`spec.yaml` apparence suivante :

```
proton: EnvironmentSpec

spec:
  codebuild_vpc_id: vpc-045ch35y28dec3a05
  codebuild_subnets:
    - subnet-04029a82e6ae46968
    - subnet-0f500a9294fc5f26a
  codebuild_security_groups:
    - sg-03bc4c4ce32d67e8d
```

AWS Proton ressources et balises

AWS Proton les ressources auxquelles est attribué un Amazon Resource Name (ARN) incluent les modèles d'environnement et leurs versions majeures et mineures, les modèles de service et leurs versions majeures et mineures, les environnements, les services, les instances de service, les composants et les référentiels. Vous pouvez baliser ces ressources pour mieux les organiser et les identifier. Vous pouvez utiliser les balises pour classer vos ressources par objectif, propriétaire, environnement ou selon d'autres critères. Pour de plus amples informations, consultez [Politiques de balisage](#). Pour suivre et gérer votre AWS Proton ressources, vous pouvez utiliser les fonctionnalités de balise décrites dans les sections suivantes.

AWS marquage

Vous pouvez attribuer des métadonnées à vos ressources AWS sous la forme d'identifications. Chaque balise est constituée d'une clé et d'une valeur facultative. Les balises peuvent vous aider à gérer, identifier, organiser, rechercher et filtrer des ressources.

Important

N'ajoutez pas de données d'identification personnelle (PII) ou d'autres informations confidentielles ou sensibles dans les étiquettes. Les étiquettes accessibles à de nombreux Services AWS, y compris la facturation. Les étiquettes ne sont pas destinées à être utilisées pour des données privées ou sensibles.

Chaque balise se compose de deux parties.

- Une clé de balise (par exemple, `CostCenter`, `Environment` ou `Project`). Les clés de balises sont sensibles à la casse.
- Une valeur de balise (facultative) (par exemple, `111122223333` ou `Production`). Les valeurs de balise sont sensibles à la casse, tout comme les clés de balise.

Les exigences d'utilisation et de désignation basiques suivantes s'appliquent aux balises.

- Chaque ressource peut avoir un maximum de 50 balises créées par l'utilisateur.

Note

Les balises créées par le système qui commencent par `aws :` les préfixes sont réservés à AWS et ne sont pas prises en compte dans cette limite. Vous ne pouvez pas modifier ou supprimer une balise commençant par le préfixe `aws :`.

- Pour chaque ressource, chaque clé d'identification doit être unique, et chaque clé d'identification peut avoir une seule valeur.
- La clé de balise doit comporter au minimum 1 caractère et au maximum 128 caractères Unicode en UTF-8.
- La valeur de balise doit être au minimum 1 caractère et au maximum 256 caractères Unicode en UTF-8.
- Les caractères autorisés dans les balises sont les lettres, les lettres, les lettres, les lettres, les lettres, les lettres, les lettres, les lettres, les lettres, les lettres, les lettres et les lettres.

AWS Protonmarquage

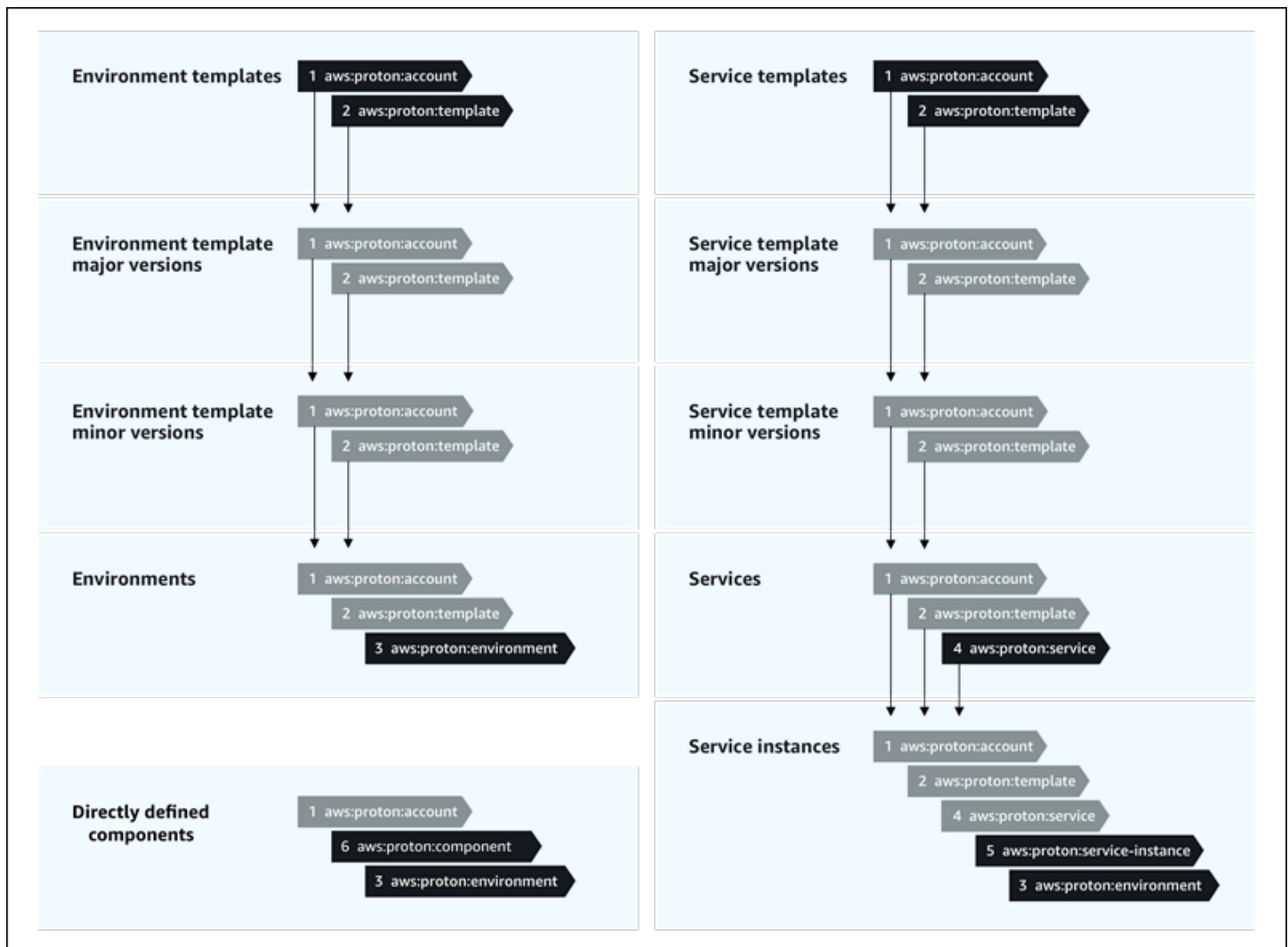
avec AWS Proton, vous pouvez utiliser à la fois les balises que vous créez et celles qui sont générées automatiquement pour vous.

AWS Proton AWSbalises gérées

Lorsque vous créez une ressource AWS Proton, AWS Proton génère automatiquement des balises gérées pour votre nouvelle ressource comme illustré dans le diagramme suivant. Les balises gérées se propagent ensuite vers d'autres ressources basées sur votre nouvelle ressource. Par exemple, les balises gérées d'un modèle d'environnement se propagent vers ses versions, et les balises gérées d'un service se propagent vers ses instances de service.

Note

Les balises gérées ne sont pas générées pour les connexions aux comptes d'environnement. Pour plus d'informations, consultez [the section called "Connexions aux comptes"](#).



Propagation des balises vers les ressources provisionnées

Si des ressources sont mises à disposition, telles que celles définies dans les modèles de service et d'environnement, prennent en chargeAWSle balisage,AWSles balises gérées se propagent sous forme de balises gérées par le client aux ressources provisionnées. Ces balises ne seront pas propagées vers une ressource provisionnée qui ne prend pas en chargeAWSbalisage.

AWS Protonapplique des balises à vos ressources enAWS Protondes comptes, des modèles enregistrés et des environnements déployés, ainsi que des services et des instances de service, comme décrit dans le tableau suivant. Vous pouvez utiliserAWSbalises gérées pour afficher et gérer vosAWS Protonressources, mais vous ne pouvez pas les modifier.

AWS clé de balise gérée	Clés gérées par le client	Description
<code>aws:proton:account</code>	<code>proton:account</code>	Le AWS compte qui crée et déploie AWS Proton ressources.
<code>aws:proton:template</code>	<code>proton:template</code>	L'ARN d'un modèle sélectionné.
<code>aws:proton:environment</code>	<code>proton:environment</code>	L'ARN d'un environnement sélectionné.
<code>aws:proton:service</code>	<code>proton:service</code>	L'ARN d'un service sélectionné.
<code>aws:proton:service-instance</code>	<code>proton:service-instance</code>	L'ARN d'une instance de service sélectionnée.
<code>aws:proton:component</code>	<code>proton:component</code>	L'ARN d'un composant sélectionné.

Voici un exemple de AWS tag géré pour un AWS Proton ressource.

```
"aws:proton:template" = "arn:aws:proton:region-id:account-id:environment-template/env-template"
```

Voici un exemple de balise gérée par le client appliquée à une ressource provisionnée et qui a été gérée par une AWS balise gérée.

```
"proton:environment:database" = "arn:aws:proton:region-id:account-id:rds/env-db"
```

avec [AWS-provisionnement géré](#), AWS Proton applique les balises propagées directement aux ressources provisionnées.

avec [provisionnement autogéré](#), AWS Proton met à disposition les balises propagées ainsi que les fichiers iAC rendus qu'il soumet dans la pull request (PR) de provisionnement. Les balises sont fournies dans la variable de type string `mapproton_tags`. Nous vous recommandons de faire référence à cette variable dans votre configuration Terraform pour inclure AWS Proton tags

default_tags. Cela se propage à toutes les ressources provisionnées.

L'exemple suivant montre cette méthode de propagation des balises dans un modèle Terraform d'environnement.

Voici le plugin proton_tags définition de la variable :

proton.environment.variables.tf :

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

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

Voici comment les valeurs de balise sont attribuées à cette variable :

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"
  }
}
```

Et voici comment vous pouvez ajouter AWS Proton des balises à votre configuration Terraform afin qu'elles soient ajoutées aux ressources provisionnées :

```
# Configure the AWS Provider
provider "aws" {
  region = var.aws_region
  default_tags {
    tags = var.proton_tags
  }
}
```

Tags gérées par les clients

Chaque AWS Proton La ressource dispose d'un quota maximum de 50 balises gérées par le client. Les tags gérés par le client sont transmis à l'enfant AWS Proton ressources de la même manière que AWS Les balises gérées le font, sauf qu'elles ne se propagent pas aux balises existantes AWS Proton aux ressources ou aux ressources mises à disposition. Si vous appliquez une nouvelle étiquette à AWS Proton ressource avec des ressources enfants existantes et si vous souhaitez que les ressources enfants existantes soient balisées avec la nouvelle balise, vous devez baliser chaque ressource enfant existante manuellement, à l'aide de la console ou AWS CLI.

Création de balises via la console et l'interface de ligne

Lorsque vous créez un AWS Proton ressource à l'aide de la console, vous avez la possibilité de créer des balises gérées par le client sur la première ou la deuxième page de la procédure de création, comme indiqué dans la capture d'écran de console suivante. Choisissez Ajouter un nouveau tag, saisissez la clé et la valeur, puis continuez.

Tags


Customer managed tags

Add tags to help you search, filter, and track your service in Proton.

Key

Value - optional

You can add up to 49 more tags.

 New tags will only propagate to service instances that you create after you have created the new tags. They won't propagate to existing service instances.

Lorsque vous avez créé une nouvelle ressource en utilisant le plugin AWS Proton console, vous pouvez afficher sa liste de balises gérées et gérées par le client depuis la page détaillée.

Créer ou modifier une balise

1. Dans [AWS Proton console](#), ouvrez un plugin AWS Proton page de détail de la ressource où vous pouvez voir une liste de balises.
2. Choisissez **Manage tags (Gérer les balises)**.
3. Dans **Gérer les balises** page, vous pouvez afficher, créer, supprimer et modifier des balises. Vous ne pouvez pas modifier les balises gérées en haut. Toutefois, vous pouvez ajouter et modifier les balises gérées par le client à l'aide de champs d'édition, répertoriés après les balises gérées.

Choisissez **Ajouter un nouveau tag** pour créer une nouvelle balise.

4. Entrez une clé et une valeur pour la nouvelle balise.
5. Pour modifier une balise, entrez une valeur dans le champ de valeur de balise pour la clé sélectionnée.
6. Pour supprimer une balise choisissez **Supprimez** pour une balise sélectionnée.
7. Lorsque vous avez terminé vos modifications, choisissez **Enregistrer les modifications**.

Créez des balises en utilisant le plugin AWS Proton AWS CLI

Vous pouvez afficher, créer, supprimer et modifier des balises à l'aide du AWS Proton AWS CLI.

Vous pouvez créer ou modifier une balise pour une ressource comme illustré dans l'exemple suivant.

```
$ aws proton tag-resource \  
  --resource-arn "arn:aws:proton:region-id:account-id:service-template/web-service" \  
  --tags '[{"key":"mykey1","value":"myval1"}, {"key":"mykey2","value":"myval2"}]'
```

Vous pouvez supprimer une balise pour une ressource comme illustré dans l'exemple suivant.

```
$ aws proton untag-resource \  
  --resource-arn "arn:aws:proton:region-id:account-id:service-template/web-service" \  
  --tag-keys ["mykey1","mykey2"]'
```

Vous pouvez répertorier les balises d'une ressource comme indiqué dans le dernier exemple.

```
$ aws proton list-tags-for-resource \  
  --resource-arn "arn:aws:proton:region-id:account-id:service-template/web-service"
```

Résolution des problèmes de AWS Proton

Découvrez comment résoudre les problèmes rencontrés avec AWS Proton.

Rubriques

- [Erreurs de déploiement faisant référence à des paramètres AWS CloudFormation dynamiques](#)

Erreurs de déploiement faisant référence à des paramètres AWS CloudFormation dynamiques

Si vous constatez des erreurs de déploiement faisant référence à vos [variables CloudFormation dynamiques](#), vérifiez qu'il s'agit d'erreurs de type [Jinja](#). Ces erreurs peuvent être causées par une mauvaise interprétation par Jinja de vos variables dynamiques. La syntaxe des paramètres CloudFormation dynamiques est très similaire à la syntaxe Jinja que vous utilisez avec vos AWS Proton paramètres.

Exemple de syntaxe de variable CloudFormation dynamique :

```
'{{resolve:secretsmanager:MySecret:SecretString:password:EXAMPLE1-90abcdef-fedc-ba987EXAMPLE}}'
```

Exemple de AWS Proton syntaxe Jinja pour un paramètre :

```
'{{ service_instance.environment.outputs.env-outputs }}'
```

Pour éviter ces erreurs d'interprétation, Jinja échappe à vos paramètres CloudFormation dynamiques, comme indiqué dans les exemples suivants.

Cet exemple provient du Guide de l'utilisateur AWS CloudFormation. Les segments AWS Secrets Manager secret-name et json-key peuvent être utilisés pour récupérer les informations de connexion stockées dans le secret.

```
MyRDSInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    DBName: 'MyRDSInstance'
    AllocatedStorage: '20'
    DBInstanceClass: db.t2.micro
```

```

Engine: mysql
MasterUsername: '{{resolve:secretsmanager:MyRDSecret:SecretString:username}}'
MasterUserPassword:
'{{resolve:secretsmanager:MyRDSecret:SecretString:password}}'

```

Pour échapper aux paramètres CloudFormation dynamiques, vous pouvez utiliser deux méthodes différentes :

- Insérez un bloc entre `{% raw %}` and `{% endraw %}` :

```

'{{% raw %}}
MyRDSInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    DBName: 'MyRDSInstance'
    AllocatedStorage: '20'
    DBInstanceClass: db.t2.micro
    Engine: mysql
    MasterUsername: '{{resolve:secretsmanager:MyRDSecret:SecretString:username}}'
    MasterUserPassword:
'{{resolve:secretsmanager:MyRDSecret:SecretString:password}}'
'{{% endraw %}}'

```

- Insérez un paramètre entre `"{{ }}"` :

```

MyRDSInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    DBName: 'MyRDSInstance'
    AllocatedStorage: '20'
    DBInstanceClass: db.t2.micro
    Engine: mysql
    MasterUsername:
"{{ '{{resolve:secretsmanager:MyRDSecret:SecretString:username}}' }}"
    MasterUserPassword:
"{{ '{{resolve:secretsmanager:MyRDSecret:SecretString:password}}' }}"

```

Pour plus d'informations, voir [Jinja s'évader](#).

Quotas AWS Proton

Le tableau suivant répertorie les quotas AWS Proton. Toutes les valeurs sont exprimées par compte AWS, par compte pris en charge AWS Région.

Quota de ressources	Limite par défaut	Ajustable?
Taille maximale du bundle de modèles	10 Mo	× Non
Taille maximale du modèle de fichier manifeste	2 Mo	× Non
Taille maximale du fichier de schéma modèle	2 Mo	× Non
Taille maximale de chaque fichier modèle	2 Mo	× Non
Longueur maximale de chaque nom de modèle	100 caractères	× Non
Nombre maximum de fichiers modèles par bundle	1	× Non
Nombre maximum de modèles enregistrés par compte, modèles de service et d'environnement combinés	1 000	✓ Oui
Nombre maximum de versions de modèle enregistrées par modèle	1 000	✓ Oui
Nombre maximum de fichiers par CodeBuildBundle de provisionnement	500	× Non
Nombre maximum d'environnements par compte	1 000	✓ Oui
Nombre maximum de services par compte	1 000	✓ Oui
Nombre maximum d'instances de service par service	20	✓ Oui
Nombre maximum de composants par compte	1 000	✓ Oui

Quota de ressources	Limite par défaut	Ajustable?
Nombre maximal de connexions à un compte d'environnement par compte d'environnement	1 000	✓ Oui

Historique du document

Le tableau suivant décrit les modifications importantes apportées à la documentation relative à la dernière version d'AWS Protonet aux commentaires des clients. Pour recevoir les notifications des mises à jour de cette documentation, abonnez-vous à un flux RSS.

- Version de l'API : 20/07/2020

Modification	Description	Date
Mise à jour de la stratégie gérée	AWSProtonCodeBuildProvisioningServiceRolePolicy la politique a été mise à jour.	12 mai 2023
Configurations de synchronisation des services.	AWS Protonajoute la prise en charge des configurations de synchronisation des services.	31 mars 2023
CodeBuild	AWS Protonajoute la prise en charge CodeBuilddu provisionnement.	16 novembre 2022
Mise à jour de la stratégie gérée	Ajout d'une AWSProtonCodeBuildProvisioningBasicAccess politique qui donne CodeBuild les autorisations nécessaires pour exécuter une version pour AWS Proton CodeBuild Provisioning.	11 novembre 2022
Propagation des balises Terraform	Ajout de la propagation des balises Terraform au chapitre sur le balisage.	16 septembre 2022

Guide de migration des API	Le guide de migration de l'API pré-GA a été supprimé.	12 août 2022
AWS Protonobjets	Ajout d'une rubrique sur AWS Proton les objets et leurs relations avec d'autres objets AWS et des objets tiers. Voir AWS Protonles objets .	29 juillet 2022
Clarifications relatives aux référentiels liés	Clarification de l'objectif des référentiels liés (enregistrés) et de leur utilisation dans l'ensemble du guide.	18 juillet 2022
Fusion de guides	Fusion des deux guides distincts de l'administrateur et de l'utilisateur en un seul guide, le Guide de AWS Proton l'utilisateur.	30 juin 2022
Mise à jour de la stratégie gérée	Politiques gérées mises à jour pour permettre l'accès aux nouvelles opérations d'AWS ProtonAPI et pour résoudre les problèmes d'autorisation liés à certaines opérations de AWS Proton console. Voir les politiques AWS gérées pour AWS Proton .	20 juin 2022
Démarrage de l'interface CLI 	Mise à jour : Démarrez AWS CLI avec le grâce à un nouveau didacticiel qui utilise la nouvelle bibliothèque de modèles.	14 juin 2022

Composants définis directement	Le chapitre Composants a été ajouté et des modifications connexes ont été apportées au guide.	1 juin 2022
AWS Protonbibliothèque de modèles	La rubrique « Bibliothèque AWS Proton de modèles » a été ajoutée.	6 mai 2022
Disponibilité générale de Terraform (GA)	Le provisionnement par pull request a été renommé en provisionnement autogéré. Ajout d'une rubrique sur les méthodes de provisionnement .	23 mars 2022
Balisage du référentiel	Ajout d'un support pour le balisage des ressources du référentiel. Consultez la section Création d'un lien vers votre référentiel .	23 mars 2022
Mise à jour de la documentation	Ajout du balisage des connexions aux comptes d'environnement.	26 novembre 2021
Synchronisation des modèles et aperçu de Terraform	Ajout du versionnement automatique des modèles avec la fonctionnalité de synchronisation des modèles pour la disponibilité générale et le provisionnement des pull requests avec Terraform en version préliminaire. Le guide de migration des API est de retour.	24 novembre 2021

mises à jour de documentation	Ajout d'un EventBridge didacticiel, d'un flux de travail de mise en route , AWS Proton d'un mode opératoire et d'améliorations apportées à la section Ensemble de modèles .	17 septembre 2021
AWS Proton Publication des panneaux d'aide de la console	Panneaux d'aide ajoutés à la console. La suppression de la version du modèle de console ne supprime plus les versions inférieures. Le guide de migration des API a été supprimé.	8 septembre 2021
AWS Proton conversion GA (GA)	Ajout d'environnements multicomptes , d'une EventBridge surveillance , de clés de condition IAM , d'une prise en charge de l'idempotence et d'une augmentation des quotas .	9 juin 2021

[Ajoutez et supprimez des instances de service pour un service et utilisez l'infrastructure externe existante pour les environnements avec AWS Proton](#)

Cette version préliminaire publique inclut des mises à jour qui vous permettent d'[ajouter et de supprimer des instances de service d'un service](#), d'[utiliser votre infrastructure externe existante dans un AWS Proton environnement](#) et d'annuler les déploiements d'environnements, d'instances de service et de pipelines. AWS Proton prend désormais en charge [PrivateLink](#). Une validation de suppression supplémentaire a été ajoutée pour empêcher qu'une version secondaire ne soit supprimée par erreur alors qu'une ressource l'utilise.

27 avril 2021

[Marquage avec AWS Proton](#)

La version préliminaire publique 2 inclut le AWS Proton [balisage](#) et la possibilité de lancer des services [sans pipeline de services](#).

5 mars 2021

[Première version](#)

La version préliminaire publique est désormais disponible dans certaines régions.

1er décembre 2020

Glossaire AWS

Pour connaître la terminologie la plus récente d'AWS, consultez le [Glossaire AWS](#) dans la Référence Glossaire AWS.

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.