Guide du développeur

AWS App Runner

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS App Runner: Guide du développeur

Copyright © 2026 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 c'est AWS App Runner ? 1
À qui s'adresse App Runner ? 1
Accès à App Runner 1
Tarification d'App Runner 2
Quelle est la prochaine étape 2
Configuration 3
Inscrivez-vous pour un Compte AWS 3
Création d’un utilisateur doté d’un accès administratif 4
Octroi d’un accès par programmation 5
Quelle est la prochaine étape 7
Prise en main 8
Conditions préalables 8
Étape 1 : créer un service App Runner 10
Étape 2 : modifiez votre code de service 20
Étape 3 : effectuer une modification de configuration 21
Étape 4 : Afficher les journaux de votre service 23
Étape 5 : nettoyer 26
Quelle est la prochaine étape 26
Architecture et concepts 28
Concepts d'App Runner 29
Configurations compatibles avec App Runner 30
Ressources pour App Runner 31
Quotas de ressources App Runner 33
Service basé sur l'image 36
Fournisseurs de référentiels d'images 36
Utilisation d'une image enregistrée dans Amazon ECR dans votre compte AWS 37
Utilisation d'une image stockée dans Amazon ECR sur un autre compte AWS 37
Utilisation d'une image stockée dans Amazon ECR Public 38
Exemple d'image 40
Service basé sur le code 41
Fournisseurs de référentiels de code source 42
Déploiement depuis votre fournisseur de référentiel de code source 42
Répertoire des sources 43
Plateformes gérées par App Runner 44

Fin du support pour les versions d'exécution gérées 44
Versions d'exécution gérées et build d'App Runner 47
En savoir plus sur les versions et la migration d'App Runner 49
Plateforme Python 53
Configuration de l'environnement d'exécution Python 54
Des légendes pour des versions d'exécution spécifiques 55
Exemples d'exécution en Python 56
Informations sur la publication 60
Plateforme Node.js 62
Configuration d'exécution de Node.js 64
Des légendes pour des versions d'exécution spécifiques 66
Exemples d'exécution de Node.js 66
Informations sur la publication 71
Plateforme Java 74
Configuration de l'environnement d'exécution Java 75
Exemples d'exécution Java 76
Informations sur la publication 80
Plateforme .NET 82
Configuration de l'environnement d'exécution .NET 84
Exemples d'exécution .NET 84
Informations sur la publication 87
Plateforme PHP 88
Configuration de l'environnement d'exécution PHP 90
Compatibilité 91
Exemples d'exécution PHP 92
Informations sur la publication 101
Plateforme Ruby 102
Configuration du runtime Ruby 104
Exemples d'exécution Ruby 104
Informations sur la publication 107
Plateforme Go 108
Configuration de l'environnement d'exécution Go 109
Exemples d'exécution Go 109
Informations sur la publication 112
Développement pour App Runner 113
Informations d'exécution 113

Directives de développement du code 115
Console App Runner 117
Disposition générale de la console 117
La page Services 118
La page du tableau de bord du service 118
La page des comptes connectés 119
La page des configurations de mise à l'échelle automatique 120
Gestion de votre service 122
Création 122
Conditions préalables 123
Créer un service 123
Reconstruire le service défaillant 139
Reconstruction d'un service App Runner défaillant à l'aide de la console App Runner 139
Reconstruction du service App Runner défaillant à l'aide de l'API App Runner ou AWS CLI 140
Déploiement 141
Méthodes de déploiement 141
Déploiement manuel 144
Configuration 146
Configurez votre service à l'aide de l'API App Runner ou AWS CLI 146
Configurez votre service à l'aide de la console App Runner 148
Configurez votre service à l'aide d'un fichier de configuration App Runner 149
Configuration de l'observabilité 149
Ressources de configuration 151
Configuration d'une surveillance de l'état 153
Connexions 155
Gérer les connexions 156
Auto scaling (Mise à l'échelle automatique) 158
Gérer le dimensionnement automatique d'un service 160
Gérez les ressources de configuration de mise à l'échelle automatique 162
noms de domaine personnalisés 170
Associer (lien) un domaine personnalisé à votre service 171
Dissocier (dissocier) un domaine personnalisé 174
Gérez des domaines personnalisés 175
Configuration d'un enregistrement d'alias Amazon Route 53 183
Suspendre/reprise 185

Comparaison entre pause et suppression 186
Lorsque votre service est suspendu 187
Suspendez et reprenez votre service 187
Suppression 189
Comparaison entre pause et suppression 189
Que supprime App Runner ? 190
Supprimer votre service 190
Variables d'environnement de référence 192
Référencement de données sensibles en tant que variables d'environnement 192
Considérations 194
Permissions 194
Gérer les variables d'environnement 196
Console App Runner 196
API App Runner ou AWS CLI 198
Réseaux 204
Terminologie 204
Conditions générales 204
Terme spécifique à la configuration du trafic sortant 205
Termes spécifiques à la configuration du trafic entrant 205
Trafic entrant 206
En-têtes 207
Activer le point de terminaison privé 207
Activer IPv6 pour les terminaux d'App Runner 220
Trafic sortant 224
Connecteur VPC 225
Sous-réseau 226
Groupe de sécurité 227
Gérer l'accès au VPC 227
Observabilité 234
Activité 234
Suivez l'activité du service App Runner 234
Journaux (CloudWatch journaux) 235
Groupes de logs et flux App Runner 236
Afficher les journaux d'App Runner dans la console 237
Métriques (CloudWatch) 240
Statistiques d'App Runner 240

Afficher les statistiques d'App Runner dans la console ... 242
Gestion des événements (EventBridge) ... 244
Création d'une EventBridge règle pour agir sur les événements App Runner ... 245
Exemples d'événements App Runner ... 245
Exemples de modèles d'événements App Runner ... 247
Référence des événements App Runner ... 248
Actions d'API (CloudTrail) ... 250
Informations sur App Runner dans CloudTrail ... 250
Comprendre les entrées du fichier journal d'App Runner ... 251
Traçage (X-Ray) ... 254
Instrumentez votre application pour le traçage ... 255
Ajoutez des autorisations X-Ray à votre rôle d'instance de service App Runner ... 259
Activez le suivi X-Ray pour votre service App Runner ... 259
Afficher les données de suivi X-Ray pour votre service App Runner ... 259
AWS WAF ACL Web ... 261
Flux de requêtes Web entrantes ... 261
Associer le Web WAF ACLs à votre service App Runner ... 262
Considérations ... 263
Permissions ... 264
Gérer le Web ACLs ... 265
Console App Runner ... 265
AWS CLI ... 269
Tester et enregistrer sur AWS WAF le Web ACLs ... 274
Fichier de configuration d'App Runner ... 276
Exemples ... 277
Exemples de fichiers de configuration ... 277
Référence ... 280
Aperçu de la structure ... 280
Section supérieure ... 281
Section de construction ... 282
Exécuter la section ... 284
API App Runner ... 288
Utilisation du AWS CLI pour travailler avec App Runner ... 288
En utilisant AWS CloudShell ... 288
Obtention des autorisations IAM pour AWS CloudShell ... 289
Interaction avec App Runner à l'aide de AWS CloudShell ... 290

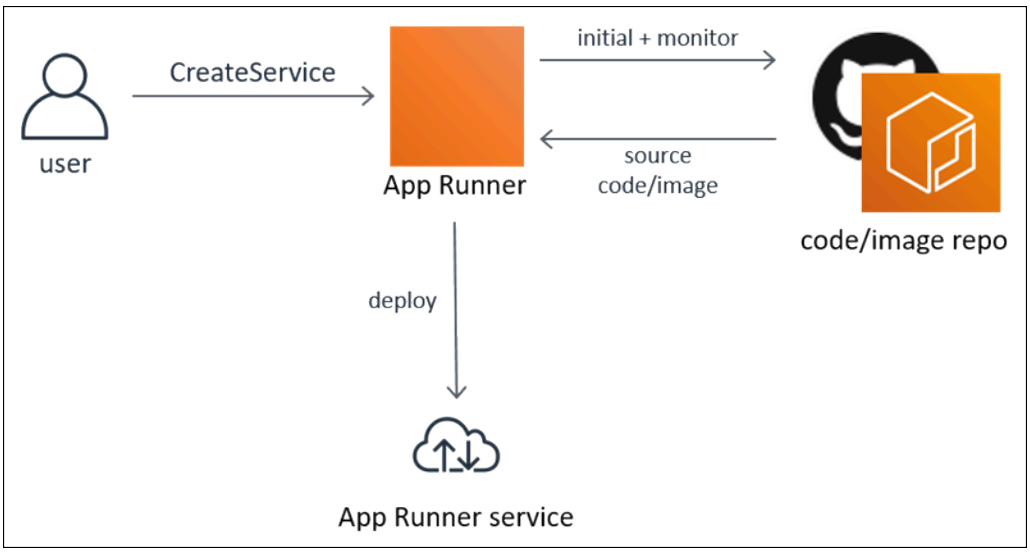
Vérification de votre service App Runner à l'aide de AWS CloudShell 293
Résolution des problèmes 294
Impossible de créer le service 294
noms de domaine personnalisés 295
Obtention d'une erreur Create Fail pour un domaine personnalisé 296
Erreur lors de l'obtention de la validation du certificat DNS en attente pour un domaine personnalisé 297
Commandes de dépannage de base 297
Renouvellement du certificat de domaine personnalisé 298
Erreur de routage de la demande 299
Erreur 404 Introuvable lors de l'envoi de HTTP/HTTPS trafic vers les points de terminaison du service App Runner 299
Échec de la connexion à Amazon RDS ou au service en aval 300
Lorsqu'il n'y a pas assez d'adresses IP pour le lancement ou le dimensionnement 303
Comment mettre à jour vos services pour en avoir davantage à votre disposition IPs 304
Calcul des IPs besoins pour vos services 304
Création de nouveaux sous-réseaux 305
Joindre des blocs CIDR secondaires à votre VPC 306
Vérification 306
Pièges courants 307
Ressources supplémentaires 308
Glossaire 308
Sécurité 309
Protection des données 310
Chiffrement des données 311
Trafic inter-réseaux 312
Gestion des identités et des accès 312
Public ciblé 313
Authentification par des identités 313
Gestion de l’accès à l’aide de politiques 314
App Runner et IAM 317
Exemples de politiques basées sur l’identité 324
Utilisation des rôles liés à un service 329
AWS politiques gérées 335
Résolution des problèmes 337
Journalisation et surveillance 338

Validation de conformité 339
Résilience 340
Sécurité de l’infrastructure 340
Points de terminaison d’un VPC 341
Configuration d'un point de terminaison VPC pour App Runner 342
Considérations relatives à la confidentialité des réseaux VPC 342
Utilisation des stratégies de point de terminaison pour contrôler l'accès avec des points de terminaison de VPC 343
Intégration au point de terminaison de l'interface 343
Modèle de responsabilité partagée 343
Images du conteneur de correctifs 343
Bonnes pratiques de sécurité 344
Bonnes pratiques de sécurité préventive 344
Bonnes pratiques de sécurité de détection 344
AWS Glossaire 346
..... cccxlvii

Qu'est-ce que c'est AWS App Runner ?

AWS App Runner est un AWS service qui fournit un moyen rapide, simple et économique de déployer à partir du code source ou d'une image de conteneur directement vers une application Web évolutive et sécurisée dans le AWS cloud. Vous n'avez pas besoin d'apprendre de nouvelles technologies, de choisir le service informatique à utiliser ou de savoir comment approvisionner et configurer les AWS ressources.

App Runner se connecte directement à votre référentiel de code ou d'images. Il fournit un pipeline d'intégration et de livraison automatique avec des opérations entièrement gérées, des performances, une évolutivité et une sécurité élevées.



À qui s'adresse App Runner ?

Si vous êtes développeur, vous pouvez utiliser App Runner pour simplifier le processus de déploiement d'une nouvelle version de votre code ou de votre référentiel d'images.

Pour les équipes opérationnelles, App Runner permet les déploiements automatiques chaque fois qu'un commit est envoyé vers le référentiel de code ou qu'une nouvelle version d'image de conteneur est envoyée vers le référentiel d'images.

Accès à App Runner

Vous pouvez définir et configurer vos déploiements de services App Runner à l'aide de l'une des interfaces suivantes :

- Console App Runner : fournit une interface Web pour gérer vos services App Runner.
- API App Runner — Fournit une RESTful API permettant d'exécuter des actions App Runner. Pour plus d'informations, veuillez consulter AWS App Runner Référence d'API .
- AWS Interface de ligne de commande (AWS CLI) : fournit des commandes pour un large éventail de AWS services, y compris Amazon VPC, et est prise en charge sous Windows, macOS et Linux. Pour de plus amples informations, veuillez consulter AWS Command Line Interface.
- AWS SDKs— Fournit des informations spécifiques à la langue APIs et prend en charge de nombreux détails de connexion, tels que le calcul des signatures, la gestion des nouvelles tentatives de demande et la gestion des erreurs. Pour de plus amples informations, veuillez consulter SDKs AWS.

Tarification d'App Runner

App Runner fournit un moyen rentable d'exécuter votre application. Vous ne payez que pour les ressources consommées par votre service App Runner. Votre service s'adapte à un nombre réduit d'instances de calcul lorsque le trafic de demandes est plus faible. Vous pouvez contrôler les paramètres d'évolutivité : le plus petit et le plus grand nombre d'instances provisionnées, et la charge la plus élevée qu'une instance gère.

Pour plus d'informations sur le dimensionnement automatique d'App Runner, consultezthe section called “Auto scaling (Mise à l'échelle automatique)”.

Pour en savoir plus sur la tarification, consultez Tarification AWS App Runner.

Quelle est la prochaine étape

Découvrez comment démarrer avec App Runner dans les rubriques suivantes :

- Configuration— Effectuez les étapes préalables à l'utilisation d'App Runner.
- Prise en main— Déployez votre première application sur App Runner.

Configuration d'App Runner

Si vous êtes un nouveau AWS client, remplissez les conditions de configuration requises répertoriées sur cette page avant de commencer à utiliser AWS App Runner.

Pour ces procédures de configuration, vous utilisez le service Gestion des identités et des accès AWS (IAM). Pour obtenir des informations complètes sur IAM, consultez les documents de référence suivants :

- Gestion des identités et des accès AWS (JE SUIS)
- Guide de l'utilisateur IAM

Inscrivez-vous pour un Compte AWS

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

Pour vous inscrire à un Compte AWS

1. Ouvrez l'https://portal.aws.amazon.com/billing/inscription.
2. Suivez les instructions en ligne.

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

   Lorsque vous vous inscrivez à un Compte AWS, un Utilisateur racine d'un compte AWSest créé. Par défaut, seul l'utilisateur racine a accès à l'ensemble des Services AWS et des ressources de ce compte. La meilleure pratique de sécurité consiste à attribuer un accès administratif à un utilisateur, et à utiliser uniquement l'utilisateur racine pour effectuer les tâches nécessitant un accès utilisateur racine.

AWS vous envoie un e-mail de confirmation une fois le processus d'inscription terminé. À tout moment, vous pouvez consulter l'activité actuelle de votre compte et gérer votre compte en accédant à https://aws.amazon.com/et en choisissant Mon compte.

Création d’un utilisateur doté d’un accès administratif

Une fois que vous vous êtes inscrit à un utilisateur administratif Compte AWS, que vous Utilisateur racine d'un compte AWS l'avez sécurisé AWS IAM Identity Center, que vous l'avez activé et que vous en avez créé un, afin de ne pas utiliser l'utilisateur root pour les tâches quotidiennes.

Sécurisez votre Utilisateur racine d'un compte AWS

1. Connectez-vous en AWS Management Consoletant que propriétaire du compte en choisissant Utilisateur root et en saisissant votre adresse Compte AWS e-mail. Sur la page suivante, saisissez votre mot de passe.

   Pour obtenir de l’aide pour vous connecter en utilisant l’utilisateur racine, consultez Connexion en tant qu’utilisateur racine dans le Guide de l’utilisateur Connexion à AWS .
2. Activez l’authentification multifactorielle (MFA) pour votre utilisateur racine.

   Pour obtenir des instructions, consultez la section Activer un périphérique MFA virtuel pour votre utilisateur Compte AWS root (console) dans le guide de l'utilisateur IAM.

Création d’un utilisateur doté d’un accès administratif

1. Activez IAM Identity Center.

   Pour obtenir des instructions, consultez Activation d’ AWS IAM Identity Center dans le Guide de l’utilisateur AWS IAM Identity Center .
2. Dans IAM Identity Center, octroyez un accès administratif à un utilisateur.

   Pour un didacticiel sur l'utilisation du Répertoire IAM Identity Center comme source d'identité, voir Configurer l'accès utilisateur par défaut Répertoire IAM Identity Center dans le Guide de AWS IAM Identity Center l'utilisateur.

Connexion en tant qu’utilisateur doté d’un accès administratif

- Pour vous connecter avec votre utilisateur IAM Identity Center, utilisez l’URL de connexion qui a été envoyée à votre adresse e-mail lorsque vous avez créé l’utilisateur IAM Identity Center.

Pour obtenir de l'aide pour vous connecter en utilisant un utilisateur d'IAM Identity Center, consultez la section Connexion au portail AWS d'accès dans le guide de l'Connexion à AWS utilisateur.

Attribution d’un accès à d’autres utilisateurs

1. Dans IAM Identity Center, créez un ensemble d’autorisations qui respecte la bonne pratique consistant à appliquer les autorisations de moindre privilège.

   Pour obtenir des instructions, consultez Création d’un ensemble d’autorisations dans le Guide de l’utilisateur AWS IAM Identity Center .
2. Attribuez des utilisateurs à un groupe, puis attribuez un accès par authentification unique au groupe.

   Pour obtenir des instructions, consultez Ajout de groupes dans le Guide de l’utilisateur AWS IAM Identity Center .

Octroi d’un accès par programmation

Les utilisateurs ont besoin d'un accès programmatique s'ils souhaitent interagir avec AWS l'extérieur du AWS Management Console. La manière d'accorder un accès programmatique dépend du type d'utilisateur qui y accède AWS.

Pour accorder aux utilisateurs un accès programmatique, choisissez l’une des options suivantes.

Quel utilisateur a besoin d’un accès programmatique ?	À	Méthode
IAM	(Recommandé) Utilisez les informations d'identification de la console comme informations d'identification temporaires pour signer les demandes programmatiques adressées au AWS CLI AWS SDKs, ou AWS APIs.	Suivez les instructions de l’interface que vous souhaitez utiliser. • Pour le AWS CLI, voir Connexion pour le développement AWS local dans le guide de AWS

Quel utilisateur a besoin d'un accès programmatique ?	À	Méthode
		Command Line Interface l'utilisateur. • Pour AWS SDKs, voir Connexion pour le développement AWS local dans le guide de référence AWS SDKs and Tools.
Identité de la main-d'œuvre (Utilisateurs gérés dans IAM Identity Center)	Utilisez des informations d'identification temporaires pour signer les demandes programmatiques adressées au AWS CLI AWS SDKs, ou AWS APIs.	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour le AWS CLI, voir Configuration du AWS CLI à utiliser AWS IAM Identity Center dans le guide de AWS Command Line Interface l'utilisateur. • Pour AWS SDKs, outils, et AWS APIs, voir Authentification IAM Identity Center dans le guide de référence AWS SDKs et Tools.
IAM	Utilisez des informations d'identification temporaires pour signer les demandes programmatiques adressées au AWS CLI AWS SDKs, ou AWS APIs.	Suivez les instructions de la section Utilisation d'informations d'identification temporaires avec AWS les ressources du Guide de l'utilisateur IAM.

Quel utilisateur a besoin d'un accès programmatique ?	À	Méthode
IAM	(Non recommandé) Utilisez des informations d'identification à long terme pour signer des demandes programmatiques adressées au AWS CLI AWS SDKs, ou AWS APIs.	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour le AWS CLI, voir Authentification à l'aide des informations d'identification utilisateur IAM dans le Guide de l'AWS Command Line Interface utilisateur. • Pour les outils AWS SDKs et, voir Authentifier à l'aide d'informations d'identif ication à long terme dans le guide de référence des outils AWS SDKs et. • Pour AWS APIs, voir Gestion des clés d'accès pour les utilisateurs IAM dans le Guide de l'utilisateur IAM.

Quelle est la prochaine étape

Vous avez effectué les étapes préalables. Pour déployer votre première application sur App Runner, consultezPrise en main.

Commencer à utiliser App Runner

AWS App Runner est un AWS service qui fournit un moyen rapide, simple et économique de transformer une image de conteneur ou un code source existant directement en un service Web actif dans le AWS Cloud.

Ce didacticiel explique comment AWS App Runner déployer votre application sur un service App Runner. Il explique comment configurer le code source et le déploiement, le build du service et le runtime du service. Il montre également comment déployer une version de code, apporter une modification de configuration et consulter les journaux. Enfin, le didacticiel montre comment nettoyer les ressources que vous avez créées en suivant les procédures du didacticiel.

Rubriques

- Conditions préalables
- Étape 1 : créer un service App Runner
- Étape 2 : modifiez votre code de service
- Étape 3 : effectuer une modification de configuration
- Étape 4 : Afficher les journaux de votre service
- Étape 5 : nettoyer
- Quelle est la prochaine étape

Conditions préalables

Avant de commencer le didacticiel, veillez à effectuer les actions suivantes :

1. Effectuez les étapes de configuration dansConfiguration.
2. Décidez si vous souhaitez travailler avec un GitHub dépôt ou un dépôt Bitbucket.
   - Pour travailler avec un Bitbucket, créez d'abord un compte Bitbucket, si vous n'en avez pas déjà un. Si vous utilisez Bitbucket pour la première fois, consultez Getting started with Bitbucket dans la documentation de Bitbucket Cloud.
   - Pour travailler avec GitHub, créez un GitHubcompte, si vous n'en avez pas déjà un. Si vous êtes nouveau GitHub dans ce domaine GitHub, consultez la section Getting started with the GitHubDocs.

> **Note**
>
> Vous pouvez créer des connexions à plusieurs fournisseurs de référentiels à partir de votre compte. Donc, si vous souhaitez effectuer un déploiement à la fois à partir d'un dépôt Bitbucket GitHub et d'un dépôt Bitbucket, vous pouvez répéter cette procédure. La prochaine fois, créez un nouveau service App Runner et créez une nouvelle connexion de compte pour l'autre fournisseur de référentiel.

3. Créez un référentiel dans votre compte fournisseur de référentiel. Ce didacticiel utilise le nom du dépôt`python-hello`. Créez des fichiers dans le répertoire racine du référentiel, avec les noms et le contenu spécifiés dans les exemples suivants.

Fichiers pour le référentiel d'**`python-hello`**exemple

Example requirements.txt

```
pyramid==2.0
```

Example server.py

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

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

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

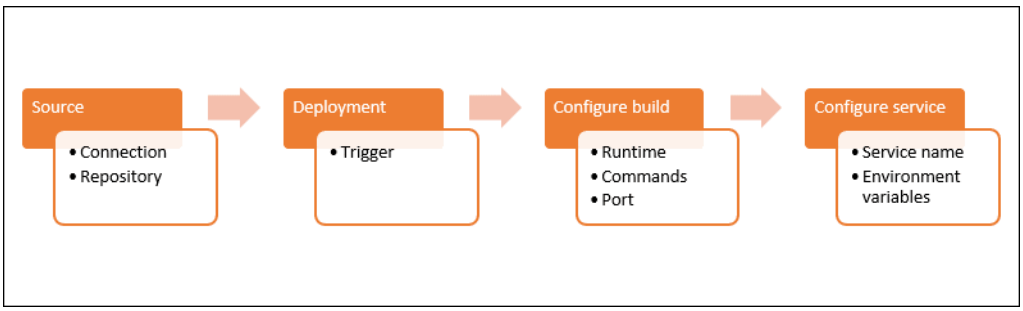
```
    server.serve_forever()
```

Étape 1 : créer un service App Runner

Au cours de cette étape, vous allez créer un service App Runner basé sur l'exemple de référentiel de code source que vous avez créé sur GitHub ou dont Bitbucket fait partiethe section called “Conditions préalables”. L'exemple contient un site Web Python simple. Voici les principales étapes à suivre pour créer un service :

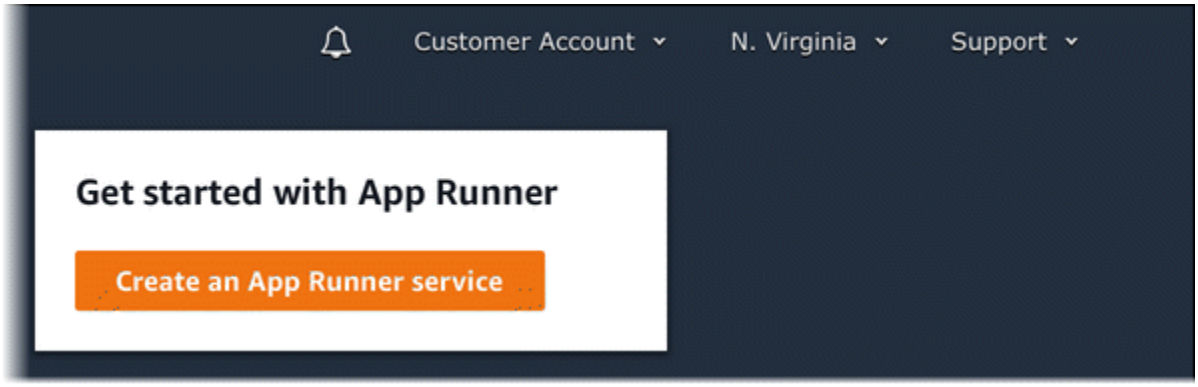
1. Configurez votre code source.
2. Configurez le déploiement de la source.
3. Configurez le build de l'application.
4. Configurez votre service.
5. Vérifiez et confirmez.

Le schéma suivant décrit les étapes de création d'un service App Runner :

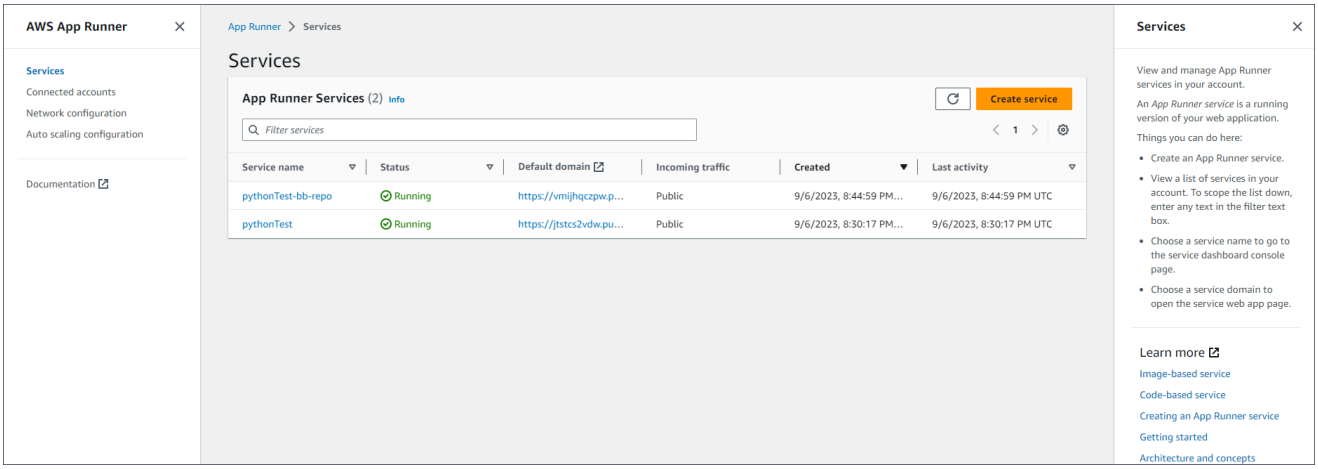


Pour créer un service App Runner basé sur un référentiel de code source

1. Configurez votre code source.
   a. Ouvrez la console App Runner, puis dans la liste des régions, sélectionnez votre Région AWS.
   b. Si aucun service App Runner Compte AWS n'est encore disponible, la page d'accueil de la console s'affiche. Choisissez Créer un service App Runner.



S'il Compte AWS possède des services existants, la page Services contenant la liste de vos services s'affiche. Choisissez Créer un service.



c. Sur la page Source et déploiement, dans la section Source, pour Type de référentiel, choisissez Référentiel de code source.

d. Sélectionnez un type de fournisseur. Choisissez l'un GitHubou l'autre ou Bitbucket.

e. Choisissez ensuite Ajouter. Si vous y êtes invité, fournissez vos informations d'identification GitHub ou celles de Bitbucket.

f. Choisissez la série d'étapes suivante en fonction du type de fournisseur que vous avez sélectionné précédemment.

> **Note**
>
> Les étapes suivantes pour installer le connecteur AWS GitHub sur votre GitHub compte sont des étapes uniques. Vous pouvez réutiliser la connexion pour créer plusieurs services App Runner basés sur les référentiels de ce compte. Lorsque vous disposez d'une connexion existante, choisissez-la et passez à la sélection du référentiel.

Il en va de même pour le connecteur AWS pour votre compte Bitbucket. Si vous utilisez les deux GitHub et Bitbucket comme référentiels de code source pour vos services App Runner, vous devez installer un connecteur AWS pour chaque fournisseur. Vous pouvez ensuite réutiliser chaque connecteur pour créer d'autres services App Runner.

- Pour GitHub, suivez ces étapes.

  i. Sur l'écran suivant, entrez un nom de connexion.

  ii. Si c'est la première fois que vous utilisez GitHub App Runner, sélectionnez Installer un autre logiciel.

  iii. Dans la boîte de GitHub dialogue AWS Connector for, si vous y êtes invité, choisissez le nom de votre GitHub compte.

  iv. Si vous êtes invité à autoriser le AWS connecteur pour GitHub, choisissez Authorize AWS Connections.

  v. Dans la boîte de GitHub dialogue Installer le AWS connecteur pour, choisissez Installer.

  Le nom de votre compte apparaît en tant que GitHub compte/organisation sélectionné. Vous pouvez désormais choisir un dépôt dans votre compte.

  vi. Pour Repository, choisissez l'exemple de référentiel que vous avez créé,`python-hello`. Pour Branch, choisissez le nom de branche par défaut de votre dépôt (par exemple, main).

  vii. Laissez le répertoire source avec la valeur par défaut. Le répertoire par défaut est la racine du dépôt. Vous avez enregistré votre code source dans le répertoire racine du référentiel lors des étapes précédentes relatives aux prérequis.

- Pour Bitbucket, procédez comme suit.

  i. Sur l'écran suivant, entrez un nom de connexion.

  ii. Si c'est la première fois que vous utilisez Bitbucket avec App Runner, sélectionnez Installer un autre.

  iii. Dans la boîte de dialogue des AWS CodeStar demandes d'accès, vous pouvez sélectionner votre espace de travail et lui accorder l'accès AWS CodeStar pour l'intégration de Bitbucket. Sélectionnez votre espace de travail, puis sélectionnez Autoriser l'accès.

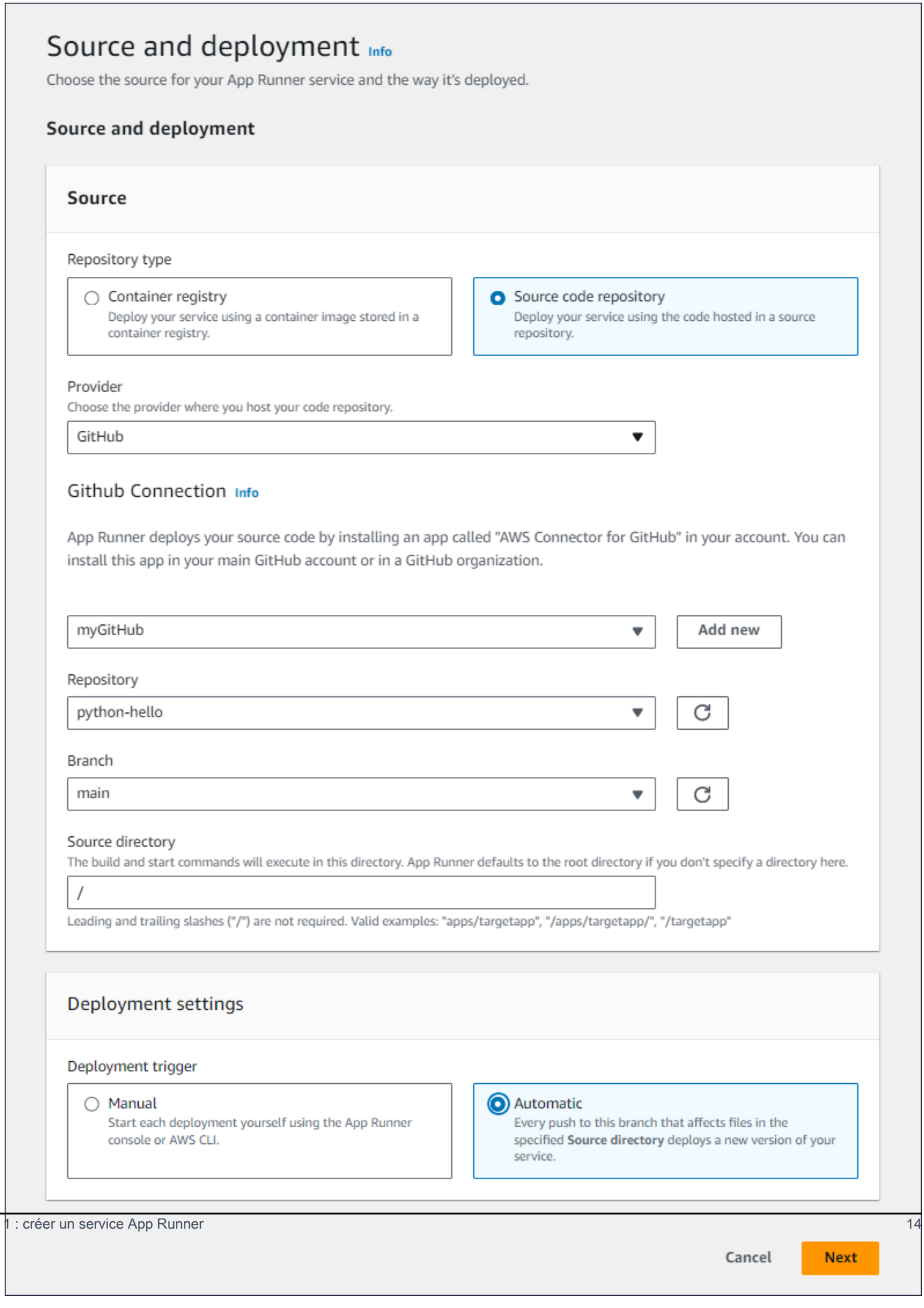
iv. Vous serez ensuite redirigé vers la AWS console. Vérifiez que l'application Bitbucket est configurée sur le bon espace de travail Bitbucket et sélectionnez Next.

v. Pour Repository, choisissez l'exemple de référentiel que vous avez créé,`python-hello`. Pour Branch, choisissez le nom de branche par défaut de votre dépôt (par exemple, main).

vi. Laissez le répertoire source avec la valeur par défaut. Le répertoire par défaut est la racine du dépôt. Vous avez enregistré votre code source dans le répertoire racine du référentiel lors des étapes précédentes relatives aux prérequis.

2. Configurez vos déploiements : dans la section Paramètres de déploiement, choisissez Automatique, puis Next.

> **Note**
>
> Avec le déploiement automatique, chaque nouvelle validation dans le répertoire source de votre référentiel déploie automatiquement une nouvelle version de votre service.


Source and deployment Info
Choose the source for your App Runner service and the way it's deployed.
Source and deployment
Source
Repository type
Container registry
Deploy your service using a container image stored in a container registry.
Source code repository
Deploy your service using the code hosted in a source repository.
Provider
Choose the provider where you host your code repository.
GitHub
Github Connection Info
App Runner deploys your source code by installing an app called "AWS Connector for GitHub" in your account. You can install this app in your main GitHub account or in a GitHub organization.
myGitHub
Add new
Repository
python-hello
Branch
main
Source directory
The build and start commands will execute in this directory. App Runner defaults to the root directory if you don't specify a directory here.
/
Leading and trailing slashes ("/") are not required. Valid examples: "apps/targetapp", "/apps/targetapp/", "/targetapp"
Deployment settings
Deployment trigger
Manual
Start each deployment yourself using the App Runner console or AWS CLI.
Automatic
Every push to this branch that affects files in the specified Source directory deploys a new version of your service.
Cancel
Next

3. Configurez le build de l'application.

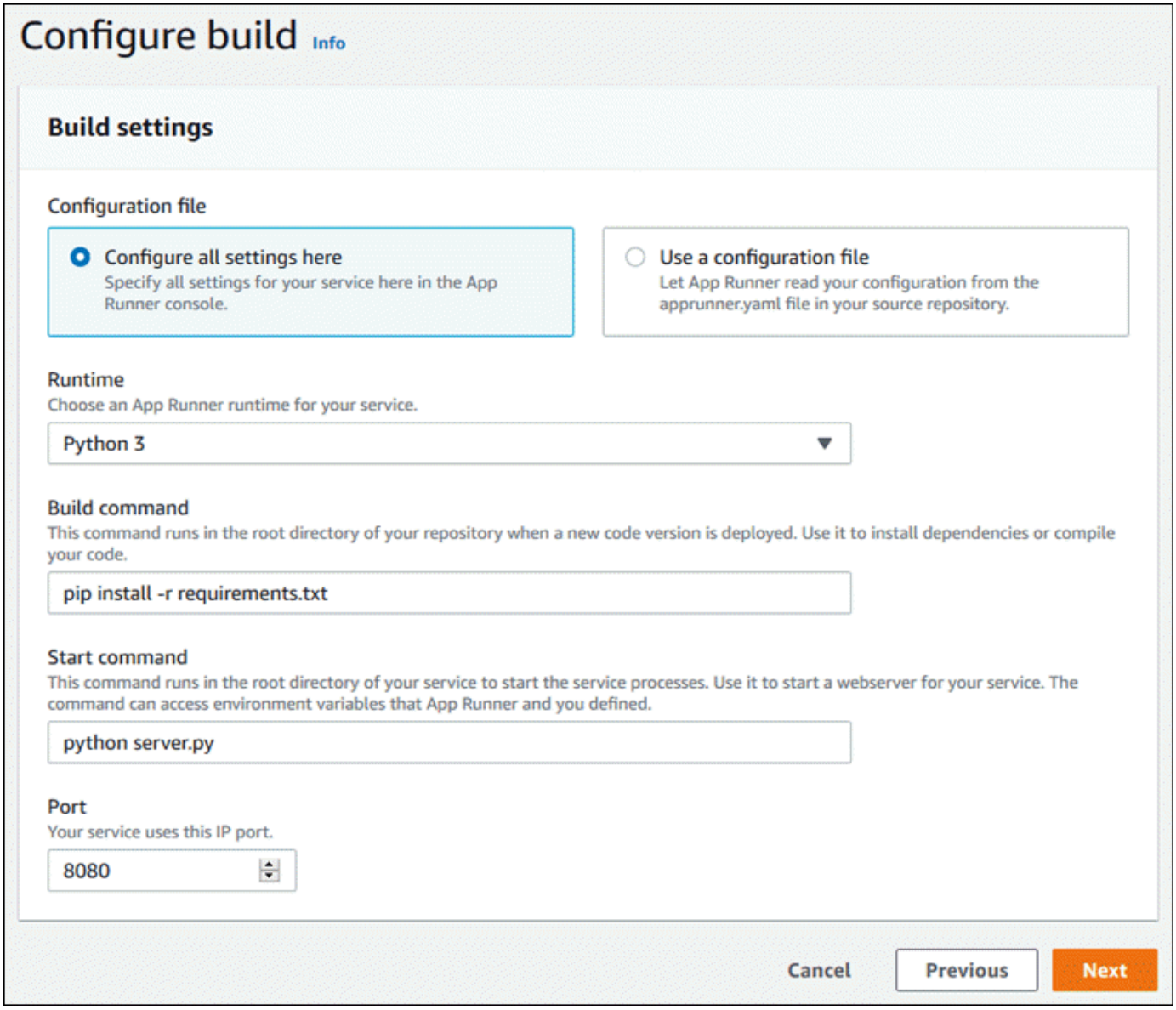
   a. Sur la page Configurer le build, dans Fichier de configuration, choisissez Configurer tous les paramètres ici.

   b. Fournissez les paramètres de compilation suivants :

      - Runtime — Choisissez Python 3.
      - Commande de construction — Entrez**`pip install -r requirements.txt`**.
      - Commande de démarrage — Entrez**`python server.py`**.
      - Port — Entrez**`8080`**.

   c. Choisissez Suivant.

> **Note**
>
> Le moteur d'exécution Python 3 crée une image Docker à l'aide d'une image Python 3 de base et de votre exemple de code Python. Il lance ensuite un service qui exécute une instance de conteneur de cette image.



4. Configurez votre service.

   a. Sur la page Configurer le service, dans la section Paramètres du service, entrez un nom de service.

   b. Sous Variables d'environnement, sélectionnez Ajouter une variable d'environnement. Fournissez les valeurs suivantes pour la variable d'environnement.

      - Source — Choisissez le texte brut
      - Nom de la variable d'environnement — **`NAME`**
      - Valeur de la variable d'environnement : n'importe quel nom (par exemple, votre prénom).

> **Note**
>
> L'application d'exemple lit le nom que vous avez défini dans cette variable d'environnement et l'affiche sur sa page Web.

c. Choisissez Suivant.

Configure service Info

Service settings

Service name

python-test

Virtual CPU & memory

1 vCPU | 2 GB

Environment variables — *optional* Info

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

No environment variables have been configured.

Add environment variable

You can add up to 50 more items.

▶ IAM policy templates

▶ Auto scaling Info

Configure automatic scaling behavior.

▶ Health check Info

Configure load balancer health checks.

▶ Security Info

Specify an Instance role and an AWS KMS encryption key

▶ Networking Info

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

▶ Observability

Configure observability tooling.

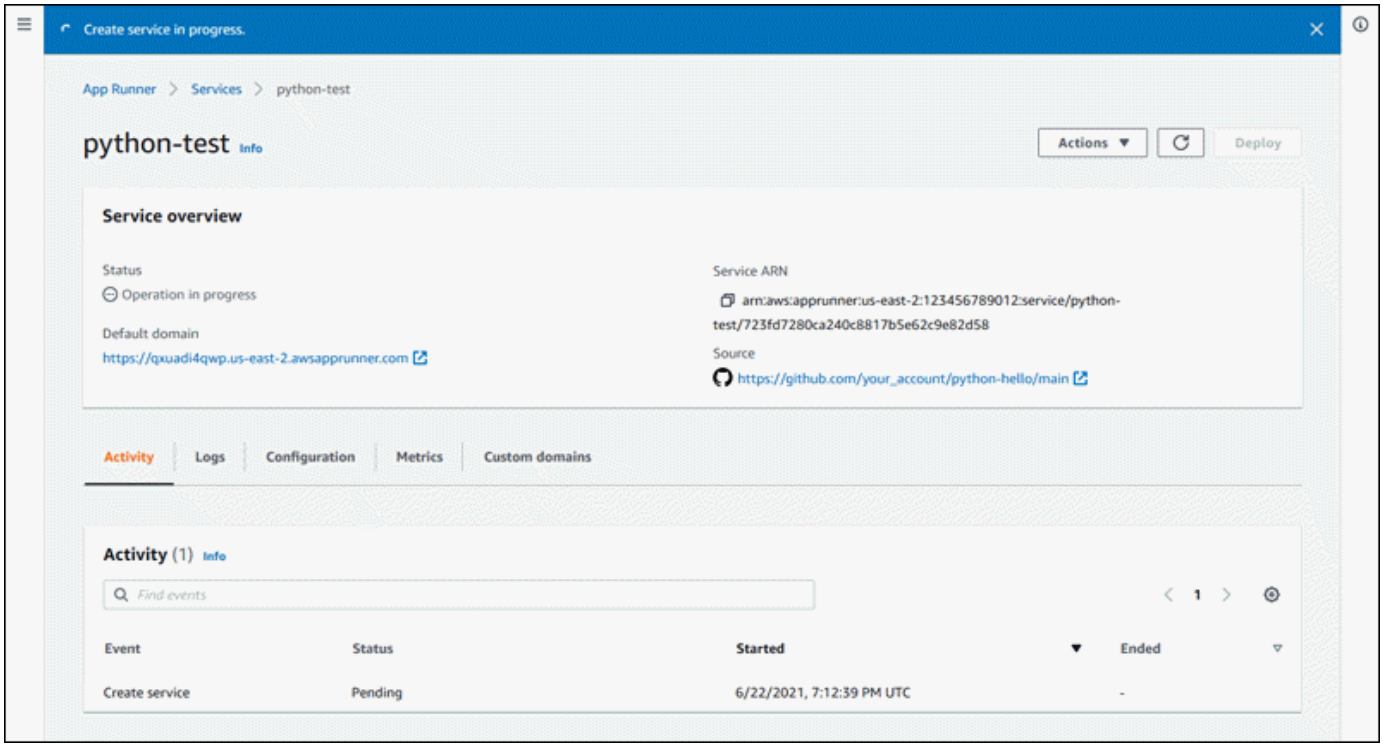
▼ Tags Info

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

Tags — *optional*

5. Sur la page Réviser et créer, vérifiez tous les détails que vous avez saisis, puis choisissez Créer et déployer.

   Si le service est créé avec succès, la console affiche le tableau de bord du service, avec une vue d'ensemble du nouveau service.



6. Vérifiez que votre service fonctionne.

   a. Sur la page du tableau de bord du service, attendez que le statut du service soit en cours d'exécution.

   b. Choisissez la valeur de domaine par défaut : il s'agit de l'URL du site Web de votre service.

> **Note**
>
> Pour renforcer la sécurité de vos applications App Runner, le domaine*.awsapprunner.com est enregistré dans la liste des suffixes publics (PSL). Pour plus de sécurité, nous vous recommandons d'utiliser des cookies avec un `__Host-` préfixe si vous devez définir des cookies sensibles dans le nom de domaine par défaut de vos applications App Runner. Cette pratique vous aidera à protéger votre domaine contre les tentatives de falsification de requêtes intersites (CSRF). Pour plus d'informations, consultez la page Set-Cookie du Mozilla Developer Network.

Une page Web affiche : Hello, ***`your name`*** !

Étape 2 : modifiez votre code de service

Au cours de cette étape, vous apportez une modification à votre code dans le répertoire source du référentiel. La CI/CD fonctionnalité App Runner crée et déploie automatiquement les modifications apportées à votre service.

Pour modifier votre code de service

1. Accédez à votre exemple de référentiel.
2. Modifiez le fichier nomméserver.py.
3. Dans l'expression affectée à la variablemessage, remplacez le texte `Hello` par`Good morning`.
4. Enregistrez et validez vos modifications dans le référentiel.
5. Les étapes suivantes illustrent la modification du code de service dans un GitHub référentiel.

    a. Accédez à votre exemple de GitHub référentiel.

    b. Choisissez le nom du fichier `server.py` pour accéder à ce fichier.

    c. Choisissez Modifier ce fichier (icône en forme de crayon).

    d. Dans l'expression affectée à la variablemessage, remplacez le texte `Hello` par`Good morning`.

e. Choisissez Valider les modifications.

6. Le nouveau commit commence à être déployé pour votre service App Runner. Sur la page du tableau de bord du service, l'état du service devient Opération en cours.

   Attendez la fin du déploiement. Sur la page du tableau de bord du service, le statut du service doit redevenir En cours d'exécution.

7. Vérifiez que le déploiement est réussi : actualisez l'onglet du navigateur où la page Web de votre service est affichée.

   La page affiche désormais le message modifié : Bonjour ***your name*** !

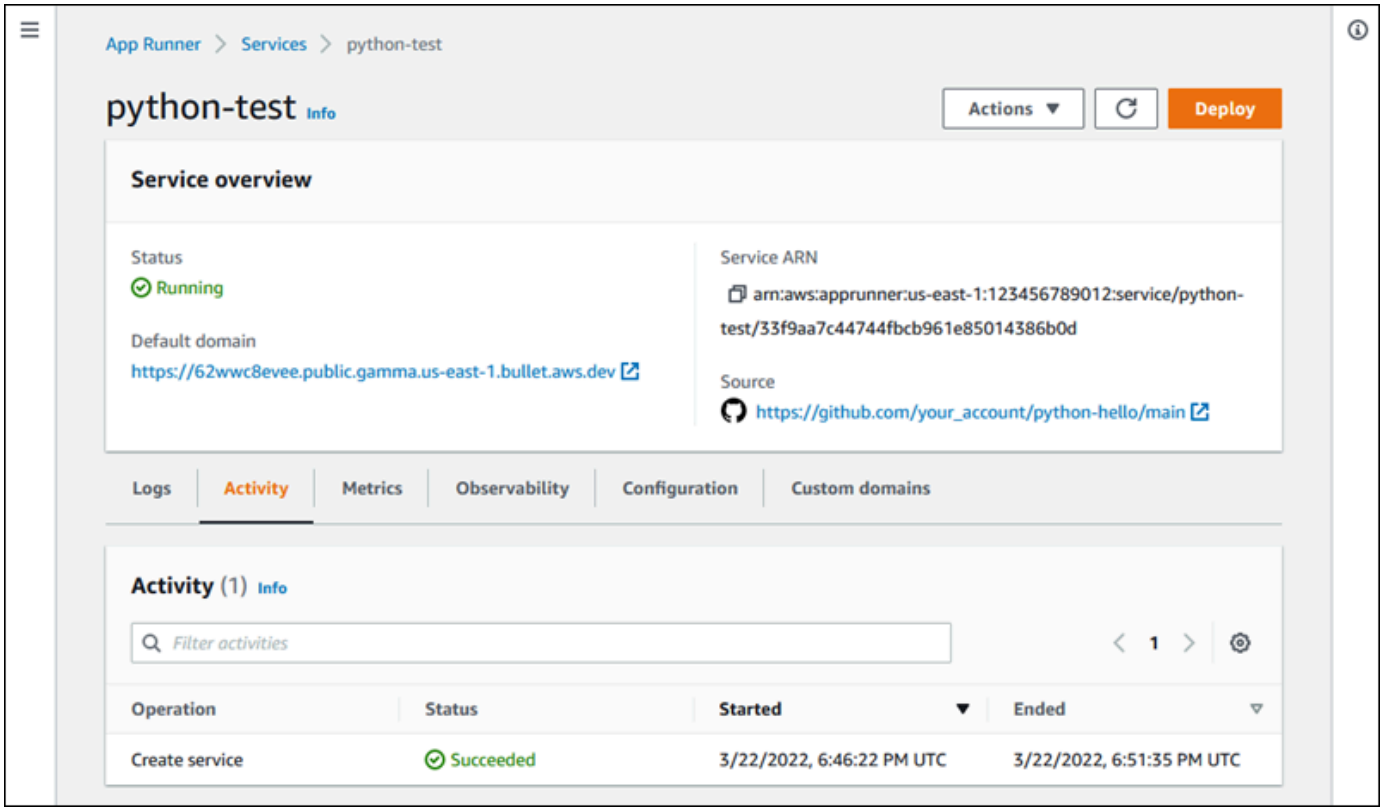
Étape 3 : effectuer une modification de configuration

Au cours de cette étape, vous modifiez la valeur de la variable d'**NAME**environnement afin de démontrer une modification de la configuration du service.

Pour modifier la valeur d'une variable d'environnement

1. Ouvrez la console App Runner, puis dans la liste des régions, sélectionnez votre Région AWS.
2. Dans le volet de navigation, choisissez Services, puis choisissez votre service App Runner.

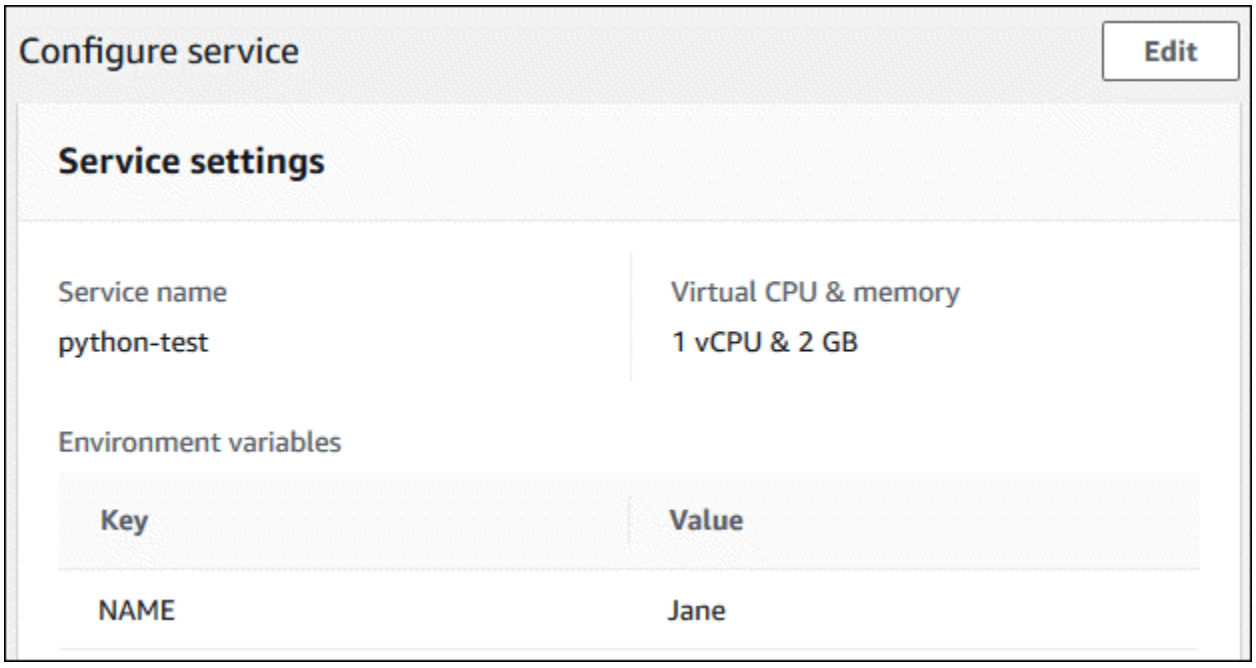
   La console affiche le tableau de bord des services avec une vue d'ensemble des services.



3. Sur la page du tableau de bord du service, choisissez l'onglet Configuration.

   La console affiche les paramètres de configuration de votre service dans plusieurs sections.

4. Dans la section Configurer le service, choisissez Modifier.



5. Pour la variable d'environnement avec la clé**`NAME`**, remplacez la valeur par un autre nom.

6. Choisissez Apply changes.

   App Runner lance le processus de mise à jour. Sur la page du tableau de bord du service, l'état du service devient Opération en cours.

7. Attendez la fin de la mise à jour. Sur la page du tableau de bord du service, le statut du service doit redevenir En cours d'exécution.

8. Vérifiez que la mise à jour est réussie : actualisez l'onglet du navigateur où la page Web de votre service est affichée.

   La page affiche désormais le nom modifié : Bonjour ***new name*** !

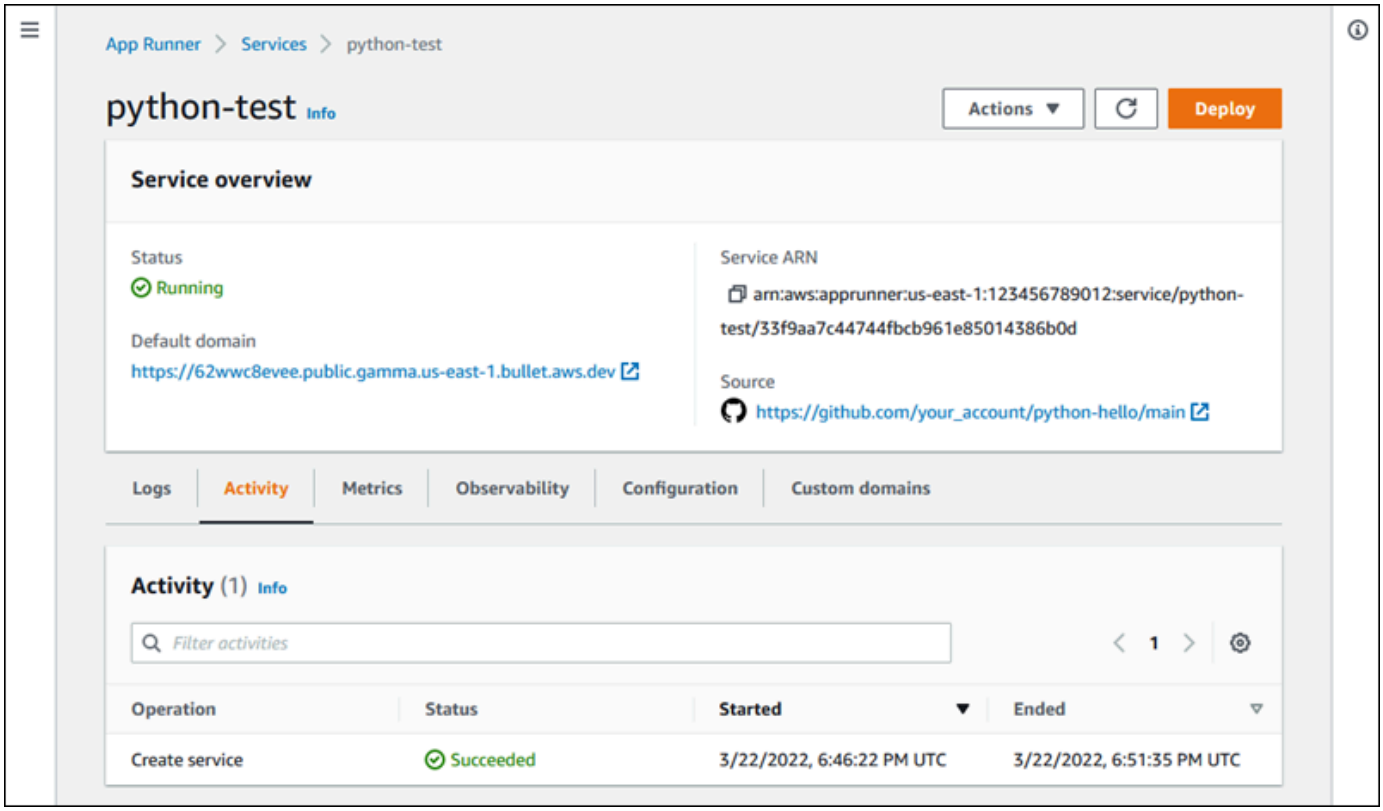
Étape 4 : Afficher les journaux de votre service

Au cours de cette étape, vous utilisez la console App Runner pour consulter les journaux de votre service App Runner. App Runner diffuse les journaux vers Amazon CloudWatch Logs (CloudWatch Logs) et les affiche sur le tableau de bord de votre service. Pour plus d'informations sur les journaux d'App Runner, consultezthe section called “Journaux (CloudWatch journaux)”.

Pour consulter les journaux de votre service

1. Ouvrez la console App Runner, puis dans la liste des régions, sélectionnez votre Région AWS.
2. Dans le volet de navigation, choisissez Services, puis choisissez votre service App Runner.

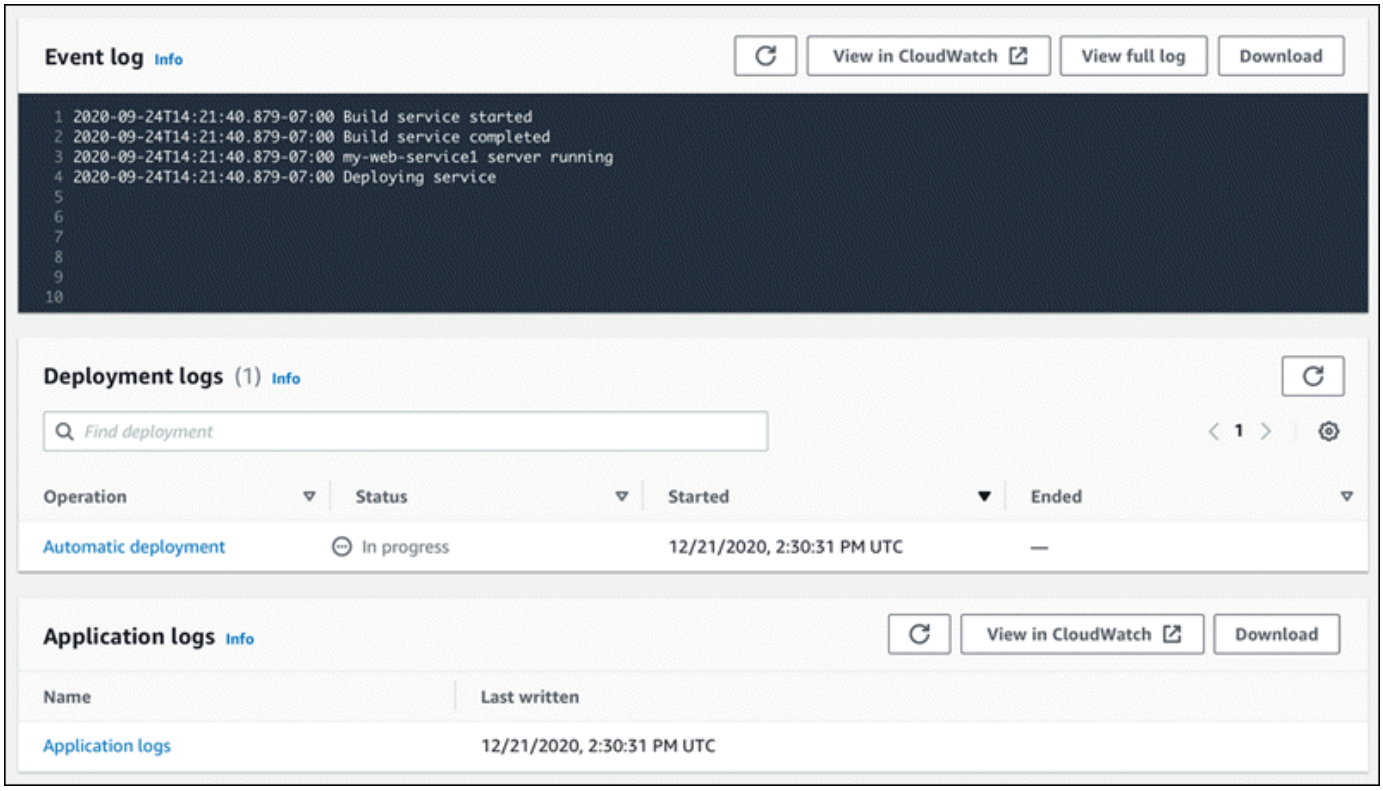
   La console affiche le tableau de bord des services avec une vue d'ensemble des services.



3. Sur la page du tableau de bord du service, choisissez l'onglet Logs.

   La console affiche quelques types de journaux dans plusieurs sections :

   - Journal des événements : activité pendant le cycle de vie de votre service App Runner. La console affiche les derniers événements.
   - Journaux de déploiement : envoyez les déploiements de référentiels à votre service App Runner. La console affiche un flux de journal distinct pour chaque déploiement.
   - Journaux d'application : sortie de l'application Web déployée sur votre service App Runner. La console combine les résultats de toutes les instances en cours d'exécution dans un seul flux de journal.



4. Pour rechercher des déploiements spécifiques, réduisez la liste des journaux de déploiement en saisissant un terme de recherche. Vous pouvez rechercher n'importe quelle valeur figurant dans le tableau.
5. Pour afficher le contenu d'un journal, choisissez Afficher le journal complet (journal des événements) ou le nom du flux du journal (journaux de déploiement et d'application).
6. Choisissez Télécharger pour télécharger un journal. Pour un flux de journal de déploiement, sélectionnez d'abord un flux de journal.
7. Choisissez Afficher dans CloudWatch pour ouvrir la CloudWatch console et utiliser toutes ses fonctionnalités pour explorer les journaux de service de votre App Runner. Pour un flux de journal de déploiement, sélectionnez d'abord un flux de journal.

> **Note**
>
> La CloudWatch console est particulièrement utile si vous souhaitez consulter les journaux d'applications d'instances spécifiques au lieu du journal d'applications combiné.

Étape 5 : nettoyer

Vous avez maintenant appris à créer un service App Runner, à consulter les journaux et à apporter des modifications. Au cours de cette étape, vous supprimez le service pour supprimer les ressources dont vous n'avez plus besoin.

Pour supprimer votre service

1. Sur la page du tableau de bord du service, choisissez Actions, puis sélectionnez Supprimer le service.
2. Dans la boîte de dialogue de confirmation, entrez le texte demandé, puis choisissez Supprimer.

   Résultat : La console accède à la page Services. Le service que vous venez de supprimer affiche le statut DELETING. Peu de temps après, il disparaît de la liste.

Pensez également à supprimer les connexions GitHub et Bitbucket que vous avez créées dans le cadre de ce didacticiel. Pour de plus amples informations, veuillez consulter the section called “Connexions”.

Quelle est la prochaine étape

Maintenant que vous avez déployé votre premier service App Runner, consultez les rubriques suivantes pour en savoir plus :

- Architecture et concepts— L'architecture, les principaux concepts et les AWS ressources liés à App Runner.
- Service basé sur l'imageet Service basé sur le code — Les deux types de sources d'applications qu'App Runner peut déployer.
- Développement pour App Runner— Ce que vous devez savoir lors du développement ou de la migration du code d'une application en vue de son déploiement vers App Runner.
- Console App Runner— Gérez et surveillez votre service à l'aide de la console App Runner.
- Gestion de votre service— Gérez le cycle de vie de votre service App Runner.
- Observabilité— Obtenez de la visibilité sur les opérations de votre service App Runner en surveillant les indicateurs, en lisant les journaux, en gérant les événements, en suivant les appels d'action du service et en suivant les événements liés aux applications tels que les appels HTTP.

- Fichier de configuration d'App Runner— Méthode basée sur la configuration pour spécifier des options pour le comportement de compilation et d'exécution de votre service App Runner.
- API App Runner— Utilisez l'interface de programmation d'applications (API) App Runner pour créer, lire, mettre à jour et supprimer des ressources App Runner.
- Sécurité— Les différentes manières de garantir AWS la sécurité du cloud lorsque vous utilisez App Runner et d'autres services.

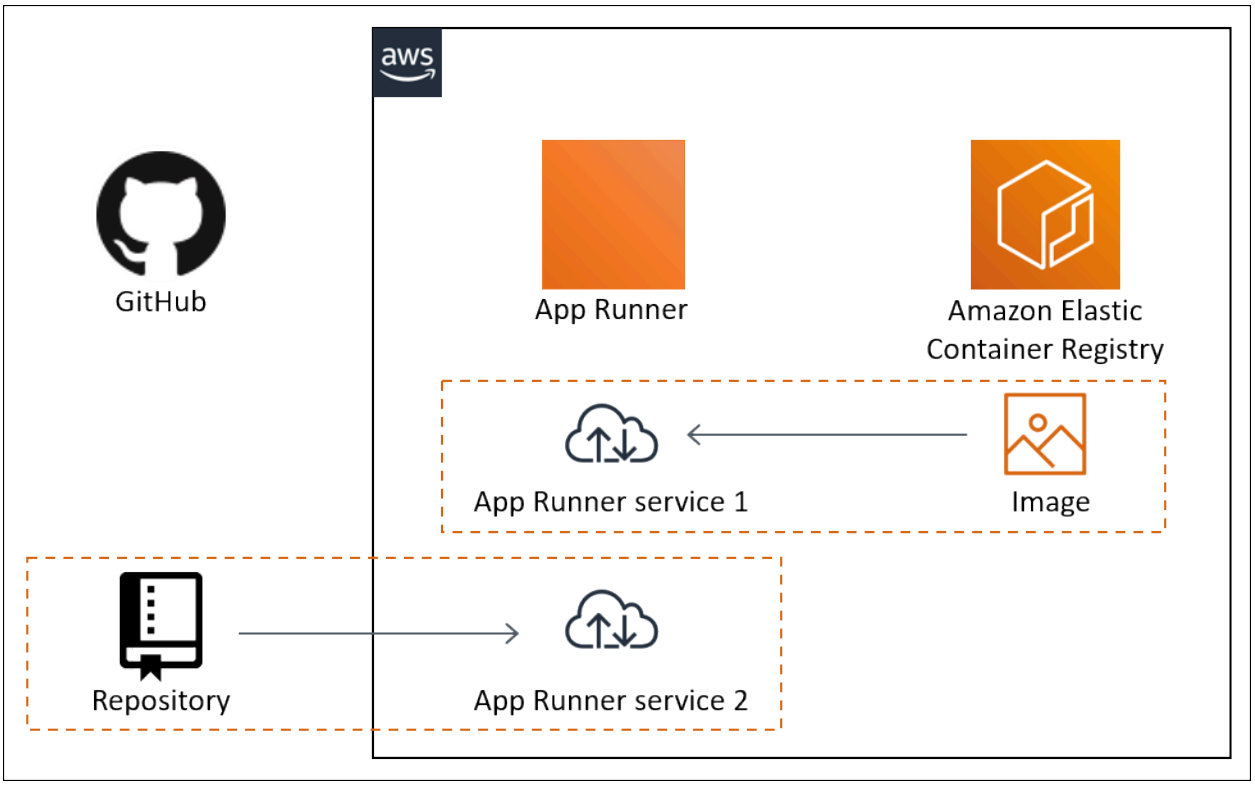
Architecture et concepts d'App Runner

AWS App Runner extrait votre code source ou votre image source d'un référentiel, et crée et gère un service Web en cours d'exécution pour vous dans le AWS Cloud. Généralement, vous n'avez besoin d'appeler qu'une seule action App Runner pour créer votre service. CreateService

Avec un référentiel d'images source, vous fournissez une image de ready-to-use conteneur qu'App Runner peut déployer pour exécuter votre service Web. Avec un référentiel de code source, vous fournissez votre code et les instructions pour créer et exécuter un service Web, et vous ciblez un environnement d'exécution spécifique. App Runner prend en charge plusieurs plateformes de programmation, chacune avec un ou plusieurs environnements d'exécution gérés pour les versions majeures de la plateforme.

À l'heure actuelle, App Runner peut récupérer votre code source depuis un Bitbucket ou un GitHubréférentiel, ou récupérer votre image source depuis Amazon Elastic Container Registry (Amazon ECR) dans votre. Compte AWS

Le schéma suivant présente une vue d'ensemble de l'architecture du service App Runner. Le schéma présente deux exemples de services : l'un déploie le code source depuis GitHub Amazon ECR et l'autre déploie une image source depuis Amazon ECR. Le même flux s'applique au référentiel Bitbucket.



Concepts d'App Runner

Les concepts clés liés à votre service Web exécuté dans App Runner sont les suivants :

- Service App Runner : AWS ressource utilisée par App Runner pour déployer et gérer votre application en fonction de son référentiel de code source ou de son image de conteneur. Un service App Runner est une version en cours d'exécution de votre application. Pour plus d'informations sur la création d'un service, consultezthe section called “Création”.
- Type de source : type de référentiel source que vous fournissez pour déployer votre service App Runner : code source ou image source.
- Fournisseur de référentiel : service de référentiel qui contient la source de votre application (par exemple GitHub, Bitbucket ou Amazon ECR).
- Connexion App Runner : AWS ressource qui permet à App Runner d'accéder au compte d'un fournisseur de référentiel (par exemple, un GitHub compte ou une organisation). Pour plus d’informations sur les connexions, consultez the section called “Connexions”.
- Runtime — Image de base pour le déploiement d'un référentiel de code source. App Runner fournit une variété de runtimes gérés pour différentes plateformes et versions de programmation. Pour de plus amples informations, veuillez consulter Service basé sur le code.
- Déploiement : action qui applique une version de votre référentiel source (code ou image) à un service App Runner. Le premier déploiement du service a lieu dans le cadre de la création du service. Les déploiements ultérieurs peuvent avoir lieu de deux manières :
  - Déploiement automatique : une CI/CD fonctionnalité. Vous pouvez configurer un service App Runner pour générer automatiquement (pour le code source) et déployer chaque version de votre application telle qu'elle apparaît dans le référentiel. Il peut s'agir d'un nouveau commit dans un référentiel de code source ou d'une nouvelle version d'image dans un référentiel d'images source.
  - Déploiement manuel : déploiement sur votre service App Runner que vous lancez explicitement.
- Domaine personnalisé : domaine que vous associez à votre service App Runner. Les utilisateurs de votre application Web peuvent utiliser ce domaine pour accéder à votre service Web au lieu du sous-domaine App Runner par défaut. Pour de plus amples informations, veuillez consulter the section called “noms de domaine personnalisés”.

> **Note**
>
> Pour renforcer la sécurité de vos applications App Runner, le domaine*.awsapprunner.com est enregistré dans la liste des suffixes publics (PSL). Pour plus de sécurité, nous vous

recommandons d'utiliser des cookies avec un `__Host-` préfixe si vous devez définir des cookies sensibles dans le nom de domaine par défaut de vos applications App Runner. Cette pratique vous aidera à protéger votre domaine contre les tentatives de falsification de requêtes intersites (CSRF). Pour plus d'informations, consultez la page Set-Cookie du Mozilla Developer Network.

- Maintenance : activité qu'App Runner effectue occasionnellement sur l'infrastructure qui exécute votre service App Runner. Lorsque la maintenance est en cours, l'état du service passe temporairement à `OPERATION_IN_PROGRESS` (Opération en cours dans la console) pendant quelques minutes. Les actions sur votre service (par exemple, déploiement, mise à jour de configuration, pause/reprise ou suppression) sont bloquées pendant cette période. Réessayez l'action quelques minutes plus tard, lorsque l'état du service revient à`RUNNING`.

  > **Note**
  >
  > Si votre action échoue, cela ne signifie pas que votre service App Runner est hors service. Votre application est active et continue de traiter les demandes. Il est peu probable que votre service soit indisponible.

  En particulier, App Runner migre votre service s'il détecte des problèmes dans le matériel sous-jacent hébergeant le service. Pour éviter toute interruption de service, App Runner déploie votre service sur un nouvel ensemble d'instances et y transfère le trafic (déploiement bleu-vert). Il est possible que vous constatiez une légère augmentation temporaire des frais.

Configurations compatibles avec App Runner

Lorsque vous configurez un service App Runner, vous spécifiez la configuration du processeur virtuel et de la mémoire à allouer à votre service. Vous payez en fonction de la configuration de calcul que vous sélectionnez. Pour plus d'informations sur la tarification, consultez Tarification Groupes de ressources AWS.

Le tableau suivant fournit des informations sur les configurations de vCPU et de mémoire prises en charge par App Runner :

CPU	Mémoire
0,25 vCPU	0,5 GO
0,25 vCPU	1 Go
0,5 vCPU	1 Go
1 vCPU	2 Go
1 vCPU	3 Go
1 vCPU	4 Go
2 vCPU	4 Go
2 vCPU	6 GO
4 vCPU	8 Go
4 vCPU	10 Go
4 vCPU	12 GO

Ressources pour App Runner

Lorsque vous utilisez App Runner, vous créez et gérez quelques types de ressources dans votre Compte AWS. Ces ressources sont utilisées pour accéder à votre code et gérer vos services.

Le tableau suivant fournit une vue d'ensemble de ces ressources :

Nom de la ressource	Description
Service	Représente une version en cours d'exécution de votre application. La majeure partie du reste de ce guide décrit les types de services, leur gestion, leur configuration et leur surveillance. ARN : `arn:aws:apprunner:` *`region`*`:`*`account-id`*` :service/`*`service-name`*` [/`*`service-id`*` ]`

Nom de la ressource	Description
Connection	Permet à vos services App Runner d'accéder à des référentiels privés stockés auprès de fournisseurs tiers. Existe en tant que ressource distincte à partager entre plusieurs services. Pour plus d'informations sur les connexions, consultez the section called “Connexions”. ARN : `arn:aws:apprunner:` *region*`:`*account-id* `:connection/` *connection-name* `[/`*connection-id* `]`
AutoScalingConfiguration	Fournit à vos services App Runner des paramètres qui contrôlent le dimensionnement automatique de votre application. Existe en tant que ressource distincte à partager entre plusieurs services. Pour plus d'informations sur la mise à l'échelle automatique, consultez the section called “Auto scaling (Mise à l'échelle automatique)”. ARN : `arn:aws:apprunner:` *region*`:`*account-id* `:autoscalingconfiguration/` *config-name* `[/`*config-revision* `[/`*config-id* `]]`
ObservabilityConfiguration	Configure des fonctionnalités supplémentaires d'observabilité des applications pour vos services App Runner. Existe en tant que ressource distincte à partager entre plusieurs services. Pour plus d'informations sur la configuration de l'observabilité, consultezthe section called “Configuration de l'observabilité”. ARN : `arn:aws:apprunner:` *region*`:`*account-id* `:observabilityconfiguration/` *config-name* `[/`*config-revision* `[/`*config-id* `]]`
VpcConnector	Configure les paramètres VPC de vos services App Runner. Existe en tant que ressource distincte à partager entre plusieurs services. Pour plus d'informations sur les fonctionnalités VPC, consultez. the section called “Trafic sortant” ARN : `arn:aws:apprunner:` *region*`:`*account-id* `:vpcconnector/` *connector-name* `[/`*connector-revision* `[/`*connector-id* `]]`

Nom de la ressource	Description
VpcIngressConnection	Il s'agit d'une AWS App Runner ressource utilisée pour configurer le trafic entrant. Il établit une connexion entre un point de terminaison d'interface VPC et le service App Runner, afin de rendre votre service App Runner accessible uniquement depuis un Amazon VPC. Pour plus d'informations sur les fonctionnalités de VPCIngress Connection, consultezthe section called “Activer le point de terminaison privé”. ARN : `arn:aws:apprunner:` *`region`*`:`*`account-id`* `:vpcingressconnection/` *`vpc-ingress-connection-name`* `[/`*`connector-id`*` ]]`

Quotas de ressources App Runner

AWS impose des quotas (également appelés limites) à votre compte pour l'utilisation AWS des ressources dans chacun d'eux Région AWS. Le tableau suivant répertorie les quotas liés aux ressources App Runner. Les quotas sont également répertoriés dans les AWS App Runner points de terminaison et les quotas dans le Références générales AWS.

Quota de ressources		Description	Valeur par défaut	Réglable ?
Services		Le nombre maximum de services que vous pouvez créer dans votre compte pour chacun d'entre eux Région AWS.	30	✓ Oui
Connections		Le nombre maximum de connexions que vous pouvez créer dans votre compte pour chacune d'entre elles Région AWS. Vous pouvez utiliser une seule connexion pour plusieurs services.	10	✓ Oui
Auto scaling configurations	names	Le nombre maximum de noms uniques que vous pouvez avoir dans les configurations de dimensionnement automatique que vous créez dans votre compte pour chacun d'entre eux	10	✓ Oui

Quota de ressources		Description	Valeur par défaut	Réglable ?
		Région AWS. Vous pouvez utiliser une seule configuration de mise à l'échelle automatique dans plusieurs services.		
	révisions par nom	Le nombre maximum de révisions de configuration de dimensionnement automatique que vous pouvez créer dans votre compte Région AWS pour chaque nom unique. Vous pouvez utiliser une seule révision de configuration d'autoscaling dans plusieurs services.	5	✗ Non
Observability configurations	names	Le nombre maximum de noms uniques que vous pouvez avoir dans les configurations d'observabilité que vous créez dans votre compte pour chacun Région AWS d'entre eux. Vous pouvez utiliser une seule configuration d'observabilité dans plusieurs services.	10	✓ Oui
	révisions par nom	Le nombre maximum de révisions de configuration d'observabilité que vous pouvez créer dans votre compte Région AWS pour chaque nom unique. Vous pouvez utiliser une seule révision de configuration d'observabilité dans plusieurs services.	10	✗ Non
VPC connectors		Le nombre maximum de connecteurs VPC que vous pouvez créer dans votre compte pour chacun d'entre eux. Région AWS Vous pouvez utiliser un seul connecteur VPC dans plusieurs services.	10	✓ Oui

Quota de ressources	Description	Valeur par défaut	Réglable ?
VPC Ingress Connection	Le nombre maximum de connexions d'entrée VPC que vous pouvez créer dans votre compte pour chacune d'elles. Région AWS Vous pouvez utiliser une seule connexion d'entrée VPC pour accéder à plusieurs services App Runner.	1	✗ Non

La plupart des quotas sont ajustables et vous pouvez demander une augmentation de quota pour eux. Pour plus d'informations, consultez Demande d'une augmentation de quota dans le Guide de l'utilisateur de Service Quotas.

Service App Runner basé sur une image source

Vous pouvez l'utiliser AWS App Runner pour créer et gérer des services basés sur deux types de source de service fondamentalement différents : le code source et l'image source. Quel que soit le type de source, App Runner se charge du démarrage, de l'exécution, de la mise à l'échelle et de l'équilibrage de charge de votre service. Vous pouvez utiliser les CI/CD fonctionnalités d'App Runner pour suivre les modifications apportées à votre image source ou à votre code. Lorsqu'App Runner découvre une modification, il crée automatiquement (pour le code source) et déploie la nouvelle version sur votre service App Runner.

Ce chapitre décrit les services basés sur une image source. Pour plus d'informations sur les services basés sur le code source, consultezService basé sur le code.

Une image source est une image de conteneur publique ou privée stockée dans un référentiel d'images. Vous pointez App Runner vers une image, et celui-ci démarre un service exécutant un conteneur basé sur cette image. Aucune étape de construction n'est nécessaire. Vous fournissez plutôt une ready-to-deploy image.

> **Note**
>
> Lorsque vous fournissez des images de conteneur, vous êtes responsable de les mettre à jour et de les corriger régulièrement. Pendant qu'App Runner gère l'infrastructure, vous devez vous assurer de la sécurité et de l' up-to-dateétat des images de conteneur fournies. Pour plus d'informations, consultez la documentation AWS App Runner

Fournisseurs de référentiels d'images

App Runner prend en charge les fournisseurs de référentiels d'images suivants :

- Amazon Elastic Container Registry (Amazon ECR) — Stocke les images privées d'un. Compte AWS
- Amazon Elastic Container Registry Public (Amazon ECR Public) : stocke des images lisibles par le public.

Cas d'utilisation des fournisseurs

- Utilisation d'une image enregistrée dans Amazon ECR dans votre compte AWS

- Utilisation d'une image stockée dans Amazon ECR sur un autre compte AWS
- Utilisation d'une image stockée dans Amazon ECR Public

Utilisation d'une image enregistrée dans Amazon ECR dans votre compte AWS

Amazon ECR stocke les images dans des référentiels. Il existe des référentiels privés et publics. Pour déployer votre image sur un service App Runner à partir d'un référentiel privé, App Runner doit être autorisé à lire votre image depuis Amazon ECR. Pour accorder cette autorisation à App Runner, vous devez attribuer un rôle d'accès à App Runner. Il s'agit d'un rôle Gestion des identités et des accès AWS (IAM) doté des autorisations d'action Amazon ECR nécessaires. Lorsque vous utilisez la console App Runner pour créer le service, vous pouvez choisir un rôle existant dans votre compte. Vous pouvez également utiliser la console IAM pour créer un nouveau rôle personnalisé. Vous pouvez également choisir que la console App Runner vous crée un rôle basé sur des politiques gérées.

Lorsque vous utilisez l'API App Runner ou le AWS CLI, vous effectuez un processus en deux étapes. Vous devez d'abord utiliser la console IAM pour créer un rôle d'accès. Vous pouvez utiliser une politique gérée fournie par App Runner ou saisir vos propres autorisations personnalisées. Ensuite, vous fournissez le rôle d'accès lors de la création du service à l'aide de l'action CreateServiceAPI.

Pour plus d'informations sur la création du service App Runner, consultezthe section called “Création”.

Utilisation d'une image stockée dans Amazon ECR sur un autre compte AWS

Lorsque vous créez un service App Runner, vous pouvez utiliser une image stockée dans un référentiel Amazon ECR appartenant à un AWS compte autre que celui dans lequel se trouve votre service. Il y a quelques considérations supplémentaires à prendre en compte lors de l'utilisation d'une image entre comptes, en plus de celles répertoriées dans la section précédente concernant une image de même compte.

- Le référentiel multi-comptes doit être associé à une politique. La politique du référentiel fournit à votre rôle d'accès les autorisations nécessaires pour lire les images du référentiel. Utilisez la politique suivante à cette fin. Remplacez le rôle dans l'`Principal`élément par le Amazon Resource Name (ARN) de votre rôle d'accès.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {"AWS": "arn:aws:iam::123456789012:role/MyApplicationRole"},
      "Action": [
        "ecr:BatchGetImage",
        "ecr:DescribeImages",
        "ecr:GetDownloadUrlForLayer"
        ],
        "Resource": "arn:aws:ecr:us-east-1:111122223333:repository/*"
    }
  ]
}
```

Pour plus d'informations sur l'attachement d'une politique de référentiel à un référentiel Amazon ECR, consultez la section Définition d'une déclaration de politique de référentiel dans le guide de l'utilisateur d'Amazon Elastic Container Registry.

- App Runner ne prend pas en charge le déploiement automatique des images Amazon ECR sur un compte différent de celui sur lequel se trouve votre service.

Utilisation d'une image stockée dans Amazon ECR Public

Amazon ECR Public stocke des images lisibles par le public. Voici les principales différences entre Amazon ECR et Amazon ECR Public que vous devez connaître dans le contexte des services App Runner :

- Les images publiques d'Amazon ECR sont lisibles par le public. Il n'est pas nécessaire de fournir un rôle d'accès lorsque vous créez un service basé sur une image Amazon ECR Public. Le référentiel n'a pas besoin d'être associé à une politique.
- App Runner ne prend pas en charge le déploiement automatique (continu) pour les images Amazon ECR Public.

Lancez un service directement depuis Amazon ECR Public

Vous pouvez lancer directement des images de conteneur d'applications Web compatibles hébergées sur la galerie publique Amazon ECR en tant que services Web exécutés sur App Runner. Lorsque vous parcourez la galerie, recherchez Launch with App Runner sur la page de la galerie pour une image. Une image avec cette option est compatible avec App Runner. Pour plus d'informations sur la galerie, consultez la section Utilisation de la galerie publique Amazon ECR dans le guide de l'utilisateur Amazon ECR Public.

Pour lancer une image de galerie en tant que service App Runner

1. Sur la page de galerie d'une image, choisissez Lancer avec App Runner.

   Résultat : La console App Runner s'ouvre dans un nouvel onglet du navigateur. La console affiche l'assistant de création de service, avec la plupart des nouveaux détails de service requis préremplis.
2. Si vous souhaitez créer votre service dans une AWS région autre que celle affichée sur la console, choisissez la région affichée dans l'en-tête de la console. Sélectionnez ensuite une autre région.
3. Pour Port, entrez le numéro de port sur lequel l'application d'imagerie écoute. Vous pouvez généralement le trouver sur la page de la galerie de l'image.
4. Modifiez éventuellement tout autre détail de configuration.
5. Choisissez Suivant, passez en revue les paramètres, puis choisissez Créer et déployer.

Exemple d'image

L'équipe App Runner conserve l'image d'hello-app-runnerexemple dans une galerie publique Amazon ECR. Vous pouvez utiliser cet exemple pour commencer à créer un service App Runner basé sur des images. Pour de plus amples informations, veuillez consulter hello-app-runner.

Service App Runner basé sur le code source

Vous pouvez l'utiliser AWS App Runner pour créer et gérer des services basés sur deux types de source de service fondamentalement différents : le code source et l'image source. Quel que soit le type de source, App Runner se charge du démarrage, de l'exécution, de la mise à l'échelle et de l'équilibrage de charge de votre service. Vous pouvez utiliser les CI/CD fonctionnalités d'App Runner pour suivre les modifications apportées à votre image source ou à votre code. Lorsqu'App Runner découvre une modification, il crée automatiquement (pour le code source) et déploie la nouvelle version sur votre service App Runner.

Ce chapitre traite des services basés sur le code source. Pour plus d'informations sur les services basés sur une image source, consultezService basé sur l'image.

Le code source est le code d'application qu'App Runner crée et déploie pour vous. Vous pointez App Runner vers un répertoire source dans un dépôt de code et vous choisissez un environnement d'exécution approprié correspondant à une version de plate-forme de programmation. App Runner crée une image basée sur l'image de base du moteur d'exécution et sur le code de votre application. Il démarre ensuite un service qui exécute un conteneur basé sur cette image.

App Runner fournit des environnements d'exécution gérés pratiques spécifiques à la plate-forme. Chacun de ces environnements d'exécution crée une image de conteneur à partir de votre code source et ajoute des dépendances de langage d'exécution à votre image. Il n'est pas nécessaire de fournir des instructions de configuration et de construction du conteneur, telles qu'un Dockerfile.

Les sous-rubriques de ce chapitre traitent des différentes plateformes prises en charge par App Runner, à savoir des plateformes gérées qui fournissent des environnements d'exécution gérés pour différents environnements de programmation et différentes versions.

Rubriques

- Fournisseurs de référentiels de code source
- Répertoire des sources
- Plateformes gérées par App Runner
- Fin du support pour les versions d'exécution gérées
- Versions d'exécution gérées et build d'App Runner
- Utilisation de la plateforme Python

- Utilisation de la plateforme Node.js
- Utilisation de la plateforme Java
- Utilisation de la plateforme .NET
- À l'aide de la plateforme PHP
- Utilisation de la plateforme Ruby
- Utilisation de la plateforme Go

Fournisseurs de référentiels de code source

App Runner déploie votre code source en le lisant depuis un dépôt de code source. App Runner prend en charge deux fournisseurs de référentiels de code source : GitHubet Bitbucket.

Déploiement depuis votre fournisseur de référentiel de code source

Pour déployer votre code source sur un service App Runner à partir d'un référentiel de code source, App Runner établit une connexion avec celui-ci. Lorsque vous utilisez la console App Runner pour créer un service, vous fournissez les détails de connexion et un répertoire des sources permettant à App Runner de déployer votre code source.

Connexions

Vous fournissez les détails de connexion dans le cadre de la procédure de création du service. Lorsque vous utilisez l'API App Runner ou le AWS CLI, une connexion constitue une ressource distincte. Tout d'abord, vous créez la connexion à l'aide de l'action CreateConnectionAPI. Vous devez ensuite fournir l'ARN de la connexion lors de la création du service à l'aide de l'action CreateServiceAPI.

Répertoire des sources

Lorsque vous créez un service, vous fournissez également un répertoire source. Par défaut, App Runner utilise le répertoire racine de votre dépôt comme répertoire source. Le répertoire source est l'emplacement de votre référentiel de code source qui stocke le code source et les fichiers de configuration de votre application. Les commandes build et start s'exécutent également à partir du répertoire source. Lorsque vous utilisez l'API App Runner ou AWS CLI pour créer ou mettre à jour un service, vous fournissez le répertoire source dans les actions d'UpdateServiceAPI CreateServiceet. Pour plus d'informations, consultez Répertoire des sourcesla section suivante.

Pour plus d'informations sur la création du service App Runner, consultezthe section called “Création”. Pour plus d'informations sur les connexions App Runner, consultezthe section called “Connexions”.

Répertoire des sources

Lorsque vous créez un service App Runner, vous pouvez fournir le répertoire source, ainsi que le référentiel et la branche. Définissez la valeur du champ Répertoire source sur le chemin du répertoire du référentiel qui stocke le code source et les fichiers de configuration de l'application. App Runner exécute les commandes de construction et de démarrage à partir du chemin du répertoire source que vous avez indiqué.

Entrez la valeur absolue du chemin du répertoire source à partir du répertoire du référentiel racine. Si vous ne spécifiez aucune valeur, il s'agit par défaut du répertoire de premier niveau du référentiel, également appelé répertoire racine du référentiel.

Vous avez également la possibilité de fournir différents chemins de répertoire source en plus du répertoire de référentiel de niveau supérieur. Cela prend en charge une architecture de référentiel monorepo, ce qui signifie que le code source de plusieurs applications est stocké dans un seul référentiel. Pour créer et prendre en charge plusieurs services App Runner à partir d'un seul monorepo, spécifiez différents répertoires sources lors de la création de chaque service.

> **Note**
>
> Si vous spécifiez le même répertoire source pour plusieurs services App Runner, les deux services seront déployés et fonctionneront individuellement.

Si vous choisissez d'utiliser un fichier `apprunner.yaml` de configuration pour définir les paramètres de votre service, placez-le dans le dossier du répertoire source du référentiel.

Si l'option Déclencheur de déploiement est définie sur Automatique, les modifications que vous validez dans le répertoire source déclencheront un déploiement automatique. Seules les modifications apportées au chemin du répertoire source déclencheront un déploiement automatique. Il est important de comprendre comment l'emplacement du répertoire source influe sur l'étendue d'un déploiement automatique. Pour plus d'informations, consultez la section Déploiements automatisés dansMéthodes de déploiement.

> **Note**
>
> Si votre service App Runner utilise les environnements d'exécution gérés par PHP et que vous souhaitez désigner un répertoire source autre que le dépôt racine par défaut, il est important d'utiliser la bonne version d'exécution de PHP. Pour de plus amples informations, veuillez consulter À l'aide de la plateforme PHP.

Plateformes gérées par App Runner

Les plateformes gérées par App Runner fournissent des environnements d'exécution gérés pour différents environnements de programmation. Chaque environnement d'exécution géré facilite la création et l'exécution de conteneurs basés sur une version d'un langage de programmation ou d'un environnement d'exécution. Lorsque vous utilisez un environnement d'exécution géré, App Runner démarre avec une image d'environnement d'exécution géré. Cette image est basée sur l'image Docker d'Amazon Linux et contient un package d'exécution de langage ainsi que des outils et des packages de dépendances populaires. App Runner utilise cette image d'exécution gérée comme image de base et ajoute le code de votre application pour créer une image Docker. Il déploie ensuite cette image pour exécuter votre service Web dans un conteneur.

Vous spécifiez un environnement d'exécution pour votre service App Runner lorsque vous créez un service à l'aide de la console App Runner ou de l'opération CreateServiceAPI. Vous pouvez également spécifier un environnement d'exécution dans le cadre de votre code source. Utilisez le `runtime` mot clé dans un fichier de configuration App Runner que vous incluez dans votre référentiel de code. La convention de dénomination d'un environnement d'exécution géré est*`<language-name><major-version>`*.

App Runner met à jour le moteur d'exécution de votre service avec la dernière version à chaque déploiement ou mise à jour de service. Si votre application nécessite une version spécifique d'un environnement d'exécution géré, vous pouvez le spécifier à l'aide du `runtime-version` mot clé dans le fichier de configuration d'App Runner. Vous pouvez verrouiller n'importe quel niveau de version, y compris une version majeure ou mineure. App Runner apporte uniquement des mises à jour de niveau inférieur à l'environnement d'exécution de votre service.

Fin du support pour les versions d'exécution gérées

Lorsque le fournisseur officiel ou la communauté d'un environnement d'exécution de langage géré déclare officiellement qu'une version est en fin de vie (EOL), App Runner déclare ensuite que le

statut de la version est « Fin de support ». Si votre service est exécuté sur une version d'exécution du langage géré dont la date de fin de support est arrivée, les politiques et recommandations suivantes s'appliquent.

Fin du support pour une version d'exécution linguistique :

- Les services existants continueront à fonctionner et à traiter le trafic même s'ils utilisent un environnement d'exécution ayant atteint la fin du Support. Toutefois, ils s'exécuteront sur des environnements d'exécution non pris en charge qui ne recevront plus de mises à jour, de correctifs de sécurité ou de support technique.
- Les mises à jour des services existants qui utilisent les environnements d'exécution de fin de support sont toujours autorisées, mais nous ne recommandons pas de continuer à utiliser les environnements d'exécution de fin de support pour un service.
- Les nouveaux services ne peuvent pas être créés à l'aide des environnements d'exécution ayant atteint la date de fin du Support.

Actions requises pour les versions d'exécution linguistiques présentant le statut de fin de support :

- Si votre service est basé sur une image source, aucune autre action de votre part n'est requise pour ce service.
- Si votre service est basé sur le code source, mettez à jour la configuration de votre service pour utiliser une version d'exécution prise en charge. Pour ce faire, sélectionnez une version d'exécution prise en charge dans la console App Runner, mettez à jour le `runtime` champ dans le fichier de configuration apprunner.yaml ou utilisez les opérations CreateService/UpdateServiceAPI ou les outils iAc pour définir le paramètre. `runtime` Pour obtenir la liste des environnements d'exécution pris en charge, consultez la page d'informations sur les versions relatives à un environnement d'exécution spécifique dans ce chapitre.
- Vous pouvez également passer à l'option de source d'image conteneur d'App Runner. Pour en savoir plus, consultez Service basé sur l'image.

> **Note**
>
> Si vous passez de Node.js 12, 14 ou 16 à Node.js 22, ou de Python 3.7 ou 3.8 à Python 3.11, sachez que Node.js 22 et Python 3.11 utilisent un processus de génération App Runner révisé qui permet des versions plus rapides et plus efficaces. Pour garantir la compatibilité

avant la mise à niveau, nous vous recommandons de consulter les instructions relatives au processus de création dans la section suivante.

Les tableaux suivants répertorient les versions d'exécution gérées par App Runner pour lesquelles une date de fin de support est définie.

Versions d'exécution	Date de fin du Support d'App Runner
Runtimes compatibles avec Python 3.8	1er décembre 2025
Runtimes compatibles avec Python 3.7	1er décembre 2025
Node.js 18 Runtimes pris en charge	1er décembre 2025
Node.js 16 Runtimes pris en charge	1er décembre 2025
Node.js 14 Runtimes pris en charge	1er décembre 2025
Node.js 12 Runtimes pris en charge	1er décembre 2025
.NET 6 *	1er décembre 2025
PHP 8.1 *	31 décembre 2025
Ruby 3.1 *	1er décembre 2025
Allez 1 *	1er décembre 2025

* App Runner ne publiera aucune nouvelle version linguistique pour les environnements d'exécution marqués d'un astérisque (*). Ces environnements d'exécution sont les suivants : .NET, PHP, Ruby et Go. Si vous avez configuré un service basé sur du code pour ces environnements d'exécution, nous vous recommandons l'une des actions suivantes :

- Le cas échéant, basculez la configuration de votre service vers un autre environnement d'exécution géré pris en charge.
- Vous pouvez également créer une image de conteneur personnalisée avec votre version d'exécution préférée et la déployer à l'aide de l'Service basé sur l'imageoption App Runner. Vous pouvez héberger votre image dans Amazon ECR.

Versions d'exécution gérées et build d'App Runner

App Runner propose un processus de génération mis à jour pour les applications qui s'exécutent sur les versions majeures les plus récentes. Ce processus de construction révisé est plus rapide et plus efficace. Il crée également une image finale moins encombrante qui contient uniquement votre code source, les artefacts de build et les temps d'exécution nécessaires à l'exécution de votre application.

Nous appelons le nouveau processus de construction la version révisée d'App Runner et le processus de construction d'origine la version originale d'App Runner. Pour éviter d'interrompre les modifications apportées aux versions antérieures des plateformes d'exécution, App Runner applique la version révisée uniquement à des versions d'exécution spécifiques, généralement des versions majeures récemment publiées.

Nous avons introduit un nouveau composant dans le fichier de `apprunner.yaml` configuration afin de rendre la version révisée rétrocompatible pour un cas d'utilisation très spécifique et de fournir plus de flexibilité pour configurer la version de votre application. Il s'agit du `pre-run`paramètre facultatif. Nous expliquons quand utiliser ce paramètre ainsi que d'autres informations utiles sur les builds dans les sections qui suivent.

Le tableau suivant indique quelle version de la version d'App Runner s'applique à des versions d'exécution gérées spécifiques. Nous continuerons à mettre à jour ce document pour vous tenir au courant de nos durées d'exécution actuelles.

Plateforme	Versions d'exécution	Processus de construction	Date de fin du Support d'App Runner
Python – Informations sur la publication	Python 3.11 (!)	Révisé	
	Python 3.8	Original	1er décembre 2025
	Python 3.7	Original	1er décembre 2025
Node.js – Informations sur la publication	Node.js 22	Révisé	
	Node.js 18	Révisé	1er décembre 2025
	Node.js 16	Original	1er décembre 2025
	Node.js 14	Original	1er décembre 2025

Plateforme	Versions d'exécution	Processus de construction	Date de fin du Support d'App Runner
	Node.js 12	Original	1er décembre 2025
Corretto — Informations sur la publication	Corretto	Original	
	Corretto 8	Original	
.NET – Informations sur la publication	.NET 6 *	Original	1er décembre 2025
PHP – Informations sur la publication	PHP 8.1 *	Original	31 décembre 2025
Ruby – Informations sur la publication	Ruby 3.1 *	Original	1er décembre 2025
Go – Informations sur la publication	Allez 1 *	Original	1er décembre 2025

Note

Certains des environnements d'exécution répertoriés incluent une date de fin de Support. Pour de plus amples informations, veuillez consulter the section called “Fin du support pour les versions d'exécution gérées”.

Important

Python 3.11 — Nous avons des recommandations spécifiques pour la configuration de build des services qui utilisent le runtime géré Python 3.11. Pour plus d'informations, consultez la rubrique relative Des légendes pour des versions d'exécution spécifiques à la plateforme Python.

En savoir plus sur les versions et la migration d'App Runner

Lorsque vous migrez votre application vers un environnement d'exécution plus récent qui utilise la version révisée, vous devrez peut-être modifier légèrement la configuration de votre version.

Pour fournir un contexte aux considérations relatives à la migration, nous allons d'abord décrire les processus de haut niveau pour la version originale d'App Runner et pour la version révisée. Nous publierons ensuite une section qui décrit les attributs spécifiques de votre service susceptibles de nécessiter des mises à jour de configuration.

La version originale d'App Runner

Le processus de création de l'application App Runner d'origine tire parti du AWS CodeBuild service. Les étapes initiales sont basées sur des images sélectionnées par le CodeBuild service. Un processus de génération de Docker suit qui utilise l'image d'exécution gérée App Runner applicable comme image de base.

Les étapes générales sont les suivantes :

1. Exécutez `pre-build` des commandes dans une CodeBuild image organisée.

   Les `pre-build` commandes sont facultatives. Ils ne peuvent être spécifiés que dans le fichier `apprunner.yaml` de configuration.
2. Exécutez les `build` commandes CodeBuild en utilisant la même image que celle de l'étape précédente.

   Les `build` commandes sont obligatoires. Ils peuvent être spécifiés dans la console App Runner, l'API App Runner ou dans le fichier `apprunner.yaml` de configuration.
3. Exécutez une version Docker pour générer une image basée sur l'image d'exécution gérée par App Runner pour votre plate-forme et votre version d'exécution spécifiques.
4. Copiez le `/app` répertoire à partir de l'image que nous avons générée à l'étape 2. La destination est l'image basée sur l'image d'exécution gérée par App Runner, que nous avons générée à l'étape 3.
5. Exécutez à nouveau les `build` commandes sur l'image d'exécution gérée par App Runner générée. Nous exécutons à nouveau les commandes de construction pour générer des artefacts de construction à partir du code source du `/app` répertoire que nous y avons copié à l'étape 4. Cette image sera ensuite déployée par App Runner pour exécuter votre service Web dans un conteneur.

Les `build` commandes sont obligatoires. Ils peuvent être spécifiés dans la console App Runner, l'API App Runner ou dans le fichier `apprunner.yaml` de configuration.

6. Exécutez `post-build` les commandes dans l' CodeBuild image de l'étape 2.

   Les `post-build` commandes sont facultatives. Ils ne peuvent être spécifiés que dans le fichier `apprunner.yaml` de configuration.

Une fois la compilation terminée, App Runner déploie l'image d'exécution gérée App Runner générée à l'étape 5 pour exécuter votre service Web dans un conteneur.

La version révisée d'App Runner

Le processus de construction révisé est plus rapide et plus efficace que le processus de construction d'origine décrit dans la section précédente. Cela élimine la duplication des commandes de compilation qui se produit dans la version précédente. Il crée également une image finale moins encombrante qui contient uniquement votre code source, les artefacts de build et les temps d'exécution nécessaires à l'exécution de votre application.

Ce processus de génération utilise une version en plusieurs étapes de Docker. Les étapes générales du processus sont les suivantes :

1. Étape de construction : lancez un processus de génération de docker qui exécute `pre-build` et `build` commande au-dessus des images de build d'App Runner.
   a. Copiez le code source de l'application `/app` dans le répertoire.

      > **Note**
      >
      > Ce `/app` répertoire est désigné comme répertoire de travail à chaque étape de la construction de Docker.

   b. Exécutez les commandes `pre-build`.

      Les `pre-build` commandes sont facultatives. Ils ne peuvent être spécifiés que dans le fichier `apprunner.yaml` de configuration.

   c. Exécutez les `build` commandes.

      Les `build` commandes sont obligatoires. Ils peuvent être spécifiés dans la console App Runner, l'API App Runner ou dans le fichier `apprunner.yaml` de configuration.

2. Étape d'empaquetage — Génère l'image finale du conteneur du client, qui est également basée sur l'image exécutée par App Runner.

   a. Copiez le `/app` répertoire de l'étape de construction précédente vers la nouvelle image Run. Cela inclut le code source de votre application et les artefacts de construction de l'étape précédente.

   b. Exécutez les `pre-run` commandes. Si vous devez modifier l'image d'exécution en dehors du `/app` répertoire à l'aide des `build` commandes, ajoutez les mêmes commandes ou les commandes obligatoires à ce segment du fichier de `apprunner.yaml` configuration.

      Il s'agit d'un nouveau paramètre qui a été introduit pour prendre en charge la version révisée d'App Runner.

      Les `pre-run` commandes sont facultatives. Ils ne peuvent être spécifiés que dans le fichier `apprunner.yaml` de configuration.

      > **Remarques**
      >
      > - Les `pre-run` commandes ne sont prises en charge que par la version révisée. Ne les ajoutez pas au fichier de configuration si votre service utilise des versions d'exécution qui utilisent la version d'origine.
      > - Si vous n'avez pas besoin de modifier quoi que ce soit en dehors du `/app` répertoire à l'aide des `build` commandes, vous n'avez pas besoin de spécifier de `pre-run` commandes.

3. Étape post-construction : cette étape reprend à partir de la phase de construction et exécute `post-build` des commandes.

   a. Exécutez les `post-build` commandes dans le `/app` répertoire.

      Les `post-build` commandes sont facultatives. Ils ne peuvent être spécifiés que dans le fichier `apprunner.yaml` de configuration.

Une fois la compilation terminée, App Runner déploie l'image Run pour exécuter votre service Web dans un conteneur.

> **Note**
>
> Ne vous laissez pas induire en erreur par les `env` entrées de la section Exécuter du `apprunner.yaml` lors de la configuration du processus de génération. Même si le paramètre de `pre-run` commande, référencé à l'étape 2 (b), se trouve dans la `env` section Exécuter, ne l'utilisez pas pour configurer votre build. Les `pre-run` commandes font uniquement référence aux `env` variables définies dans la section Build du fichier de configuration. Pour plus d'informations, consultez le Exécuter la section chapitre sur le fichier de configuration d'App Runner.

Exigences de service pour la prise en compte de la migration

Si votre environnement d'application répond à l'une de ces deux exigences, vous devrez revoir la configuration de votre build en ajoutant des `pre-run` commandes.

- Si vous devez modifier quoi que ce soit en dehors du `/app` répertoire à l'aide des `build` commandes.
- Si vous devez exécuter les `build` commandes deux fois pour créer l'environnement requis. Il s'agit d'une exigence très inhabituelle. La grande majorité des versions ne le feront pas.

Modifications en dehors du **`/app`** répertoire

- La version révisée d'App Runner part du principe que votre application ne possède aucune dépendance en dehors du `/app` répertoire.
- Les commandes que vous fournissez avec le `apprunner.yaml` fichier, l'API App Runner ou la console App Runner doivent générer des artefacts de build dans le `/app` répertoire.
- Vous pouvez modifier les `post-build` commandes `pre-buildbuild`, et pour vous assurer que tous les artefacts de construction se trouvent dans le `/app` répertoire.
- Si votre application a besoin du build pour modifier davantage l'image générée pour votre service, en dehors du `/app` répertoire, vous pouvez utiliser les nouvelles `pre-run` commandes du`apprunner.yaml`. Pour de plus amples informations, veuillez consulter Configuration des options du service App Runner à l'aide d'un fichier de configuration.

Exécution des **`build`** commandes deux fois

- La version originale d'App Runner exécute les `build` commandes deux fois, d'abord à l'étape 2, puis de nouveau à l'étape 5. La version révisée d'App Runner remédie à cette redondance et n'exécute les `build` commandes qu'une seule fois. Si votre application doit exceptionnellement exiger que `build` les commandes s'exécutent deux fois, la version révisée d'App Runner offre la possibilité de spécifier et d'exécuter à nouveau les mêmes commandes à l'aide du `pre-run` paramètre. Cela permet de conserver le même comportement de double construction.

Utilisation de la plateforme Python

> **Important**
>
> App Runner mettra fin au support de Python 3.7 et Python 3.8 le 1er décembre 2025. Pour des recommandations et de plus amples informations, consultezthe section called “Fin du support pour les versions d'exécution gérées”.

La plateforme AWS App Runner Python fournit des environnements d'exécution gérés. Chaque environnement d'exécution facilite la création et l'exécution de conteneurs avec des applications Web basées sur une version de Python. Lorsque vous utilisez un environnement d'exécution Python, App Runner démarre avec une image d'exécution Python gérée. Cette image est basée sur l'image Docker d'Amazon Linux et contient le package d'exécution d'une version de Python ainsi que certains outils et packages de dépendances populaires. App Runner utilise cette image d'exécution gérée comme image de base et ajoute le code de votre application pour créer une image Docker. Il déploie ensuite cette image pour exécuter votre service Web dans un conteneur.

Vous spécifiez un environnement d'exécution pour votre service App Runner lorsque vous créez un service à l'aide de la console App Runner ou de l'opération CreateServiceAPI. Vous pouvez également spécifier un environnement d'exécution dans le cadre de votre code source. Utilisez le `runtime` mot clé dans un fichier de configuration App Runner que vous incluez dans votre référentiel de code. La convention de dénomination d'un environnement d'exécution géré est*`<language-name><major-version>`*.

Pour les noms et versions d'exécution Python valides, consultezthe section called “Informations sur la publication”.

App Runner met à jour le moteur d'exécution de votre service avec la dernière version à chaque déploiement ou mise à jour de service. Si votre application nécessite une version spécifique d'un

environnement d'exécution géré, vous pouvez le spécifier à l'aide du `runtime-version` mot clé dans le fichier de configuration d'App Runner. Vous pouvez verrouiller n'importe quel niveau de version, y compris une version majeure ou mineure. App Runner apporte uniquement des mises à jour de niveau inférieur à l'environnement d'exécution de votre service.

Syntaxe de version pour les environnements d'exécution Python : `major[.minor[.patch]]`

Par exemple : `3.8.5`

Les exemples suivants illustrent le verrouillage des versions :

- `3.8`— Verrouillez les versions majeures et mineures. App Runner met à jour uniquement les versions de correctif.
- `3.8.5`— Verrouillez vers une version de correctif spécifique. App Runner ne met pas à jour votre version d'exécution.

Rubriques

- Configuration de l'environnement d'exécution Python
- Des légendes pour des versions d'exécution spécifiques
- Exemples d'exécution en Python
- Informations sur la version de Python Runtime

Configuration de l'environnement d'exécution Python

Lorsque vous choisissez un environnement d'exécution géré, vous devez également configurer, au minimum, créer et exécuter des commandes. Vous les configurez lors de la création ou de la mise à jour de votre service App Runner. Pour ce faire, vous pouvez utiliser l'une des méthodes suivantes :

- Utilisation de la console App Runner : spécifiez les commandes dans la section Configurer le build de l'onglet Processus de création ou configuration.
- Utilisation de l'API App Runner : appelez l'opération CreateServiceou UpdateServiceAPI. Spécifiez les commandes à l'aide des `StartCommand` membres `BuildCommand` et du type de CodeConfigurationValuesdonnées.
- Utilisation d'un fichier de configuration : spécifiez une ou plusieurs commandes de génération en trois phases de génération au maximum, ainsi qu'une seule commande d'exécution servant à démarrer votre application. Il existe d'autres paramètres de configuration facultatifs.

La fourniture d'un fichier de configuration est facultative. Lorsque vous créez un service App Runner à l'aide de la console ou de l'API, vous spécifiez si App Runner obtient vos paramètres de configuration directement lors de sa création ou à partir d'un fichier de configuration.

Des légendes pour des versions d'exécution spécifiques

> **Note**
>
> App Runner exécute désormais un processus de génération mis à jour pour les applications basé sur les versions d'exécution suivantes : Python 3.11, Node.js 22 et Node.js 18. Si votre application s'exécute sur l'une de ces versions d'exécution, consultez Versions d'exécution gérées et build d'App Runner pour plus d'informations sur le processus de génération révisé. Les applications qui utilisent toutes les autres versions d'exécution ne sont pas affectées et continuent à utiliser le processus de génération d'origine.

Python 3.11 (version révisée d'App Runner)

Utilisez les paramètres suivants dans le fichier apprunner.yaml pour le runtime géré de Python 3.11.

- Réglez la `runtime` clé dans la section supérieure sur `python311`

  Example

  ```
  runtime: python311
  ```

- Utilisez le `pip3` au lieu de `pip` pour installer les dépendances.
- Utilisez l'`python3`interpréteur au lieu de`python`.
- Exécutez le `pip3` programme d'installation sous forme de `pre-run` commande. Python installe les dépendances en dehors du `/app` répertoire. Comme App Runner exécute la version révisée d'App Runner pour Python 3.11, tout ce qui est installé en dehors du `/app` répertoire par le biais de commandes dans la section Build du `apprunner.yaml` fichier sera perdu. Pour de plus amples informations, veuillez consulter La version révisée d'App Runner.

  Example

  ```
  run:
    runtime-version: 3.11
    pre-run:
  ```

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

Pour plus d'informations, consultez également l'exemple d'un fichier de configuration étendu pour Python 3.11 plus loin dans cette rubrique.

Exemples d'exécution en Python

Les exemples suivants montrent les fichiers de configuration App Runner permettant de créer et d'exécuter un service Python. Le dernier exemple est le code source d'une application Python complète que vous pouvez déployer sur un service d'exécution Python.

> **Note**
>
> La version d'exécution utilisée dans ces exemples est `3.7.7` et`3.11`. Vous pouvez le remplacer par la version que vous souhaitez utiliser. Pour la dernière version d'exécution de Python prise en charge, consultezthe section called “Informations sur la publication”.

Fichier de configuration Python minimal

Cet exemple montre un fichier de configuration minimal que vous pouvez utiliser avec un environnement d'exécution géré en Python. Pour les hypothèses formulées par App Runner avec un fichier de configuration minimal, consultezthe section called “Exemples de fichiers de configuration”.

Python 3.11 utilise les `python3` commandes `pip3` et. Pour plus d'informations, consultez l'exemple d'un fichier de configuration étendu pour Python 3.11 plus loin dans cette rubrique.

Example apprunner.yaml

```
version: 1.0
runtime: python3
build:
  commands:
    build:
      - pip install pipenv
      - pipenv install
```

```
run:
  command: python app.py
```

Fichier de configuration Python étendu

Cet exemple montre l'utilisation de toutes les clés de configuration avec un environnement d'exécution géré par Python.

> **Note**
>
> La version d'exécution utilisée dans ces exemples est*3.7.7*. Vous pouvez le remplacer par la version que vous souhaitez utiliser. Pour la dernière version d'exécution de Python prise en charge, consultezthe section called “Informations sur la publication”.
> Python 3.11 utilise les `python3` commandes `pip3` et. Pour plus d'informations, consultez l'exemple d'un fichier de configuration étendu pour Python 3.11 plus loin dans cette rubrique.

Example apprunner.yaml

```
version: 1.0
runtime: python3
build:
  commands:
    pre-build:
      - wget -c https://s3.amazonaws.com/amzn-s3-demo-bucket/test-lib.tar.gz -O - | tar
 -xz
    build:
      - pip install pipenv
      - pipenv install
    post-build:
      - python manage.py test
  env:
    - name: DJANGO_SETTINGS_MODULE
      value: "django_apprunner.settings"
    - name: MY_VAR_EXAMPLE
      value: "example"
run:
  runtime-version: 3.7.7
  command: pipenv run gunicorn django_apprunner.wsgi --log-file -
  network:
    port: 8000
    env: MY_APP_PORT
```

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

Fichier de configuration Python étendu — Python 3.11 (utilise une version révisée)

Cet exemple montre l'utilisation de toutes les clés de configuration avec un environnement d'exécution géré Python 3.11 dans le`apprunner.yaml`. Cet exemple inclut une `pre-run` section, car cette version de Python utilise la version révisée d'App Runner.

Le `pre-run` paramètre n'est pris en charge que par la version révisée d'App Runner. N'insérez pas ce paramètre dans votre fichier de configuration si votre application utilise des versions d'exécution prises en charge par la version originale d'App Runner. Pour de plus amples informations, veuillez consulter Versions d'exécution gérées et build d'App Runner.

> **Note**
>
> La version d'exécution utilisée dans ces exemples est*3.11*. Vous pouvez le remplacer par la version que vous souhaitez utiliser. Pour la dernière version d'exécution de Python prise en charge, consultezthe section called “Informations sur la publication”.

Example apprunner.yaml

```
version: 1.0
runtime: python311
build:
  commands:
    pre-build:
      - wget -c https://s3.amazonaws.com/amzn-s3-demo-bucket/test-lib.tar.gz -O - | tar
 -xz
    build:
      - pip3 install pipenv
```

```
      - pipenv install
    post-build:
      - python3 manage.py test
  env:
    - name: DJANGO_SETTINGS_MODULE
      value: "django_apprunner.settings"
    - name: MY_VAR_EXAMPLE
      value: "example"
run:
  runtime-version: 3.11
  pre-run:
    - pip3 install pipenv
    - pipenv install
    - python3 copy-global-files.py
  command: pipenv run gunicorn django_apprunner.wsgi --log-file -
  network:
    port: 8000
    env: MY_APP_PORT
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
  secrets:
    - name: my-secret
      value-from: "arn:aws:secretsmanager:us-
east-1:123456789012:secret:testingstackAppRunnerConstr-kJFXde2ULKbT-S7t8xR:username::"
    - name: my-parameter
      value-from: "arn:aws:ssm:us-east-1:123456789012:parameter/parameter-name"
    - name: my-parameter-only-name
      value-from: "parameter-name"
```

Source complète de l'application Python

Cet exemple montre le code source d'une application Python complète que vous pouvez déployer sur un service d'exécution Python.

Example requirements.txt

```
pyramid==2.0
```

Example server.py

```
from wsgiref.simple_server import make_server
from pyramid.config import Configurator
```

```
from pyramid.response import Response
import os

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

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

Example apprunner.yaml

```
version: 1.0
runtime: python3
build:
  commands:
    build:
      - pip install -r requirements.txt
run:
  command: python server.py
```

Informations sur la version de Python Runtime

> **Important**
>
> App Runner mettra fin au support de Python 3.7 et Python 3.8 le 1er décembre 2025. Pour des recommandations et de plus amples informations, consultezthe section called “Fin du support pour les versions d'exécution gérées”.

Cette rubrique répertorie tous les détails des versions d'exécution de Python prises en charge par App Runner.

Versions d'exécution prises en charge — version révisée d'App Runner

Nom de l’exécution	Versions mineures	Forfaits inclus
Python 3.11 (python311)	3,1,114	SQLite 3,50,2
	3,1,113	SQLite 3,50,2
	3,1,113	SQLite 3,50,1
	3,1,112	SQLite 3,50,0
	3.11,11	SQLite 3,49.1
	3,1,110	SQLite 3,46.1
	3.11.9	SQLite 3,46.1
	3.11.8	SQLite 3,45,2
	3.11.7	SQLite 3,44,2

Remarques

- Python 3.11 — Nous avons des recommandations spécifiques pour la configuration de build des services qui utilisent le runtime géré Python 3.11. Pour plus d'informations, consultez la rubrique consacrée Des légendes pour des versions d'exécution spécifiques à la plateforme Python.
- App Runner fournit un processus de compilation révisé pour des environnements d'exécution majeurs spécifiques qui ont été publiés plus récemment. Pour cette raison, vous verrez des références à la version révisée d'App Runner et à la version originale d'App Runner dans certaines sections de ce document. Pour plus d'informations, consultez Versions d'exécution gérées et build d'App Runner.

Versions d'exécution prises en charge : version originale d'App Runner

Nom de l’exécution	Versions mineures	Forfaits inclus
Python 3 (python3)	3,8,20	SQLite 3,50,2
	3,8,20	SQLite 3,50,1
	3,8,20	SQLite 3,50,0
	3,8,16	SQLite 3,46.1
	3,8,15	SQLite 3,40,0
	3.7,16	SQLite 3,50,2
	3.7,16	SQLite 3,50,0
	3,7,15	SQLite 3,40,0
	3.7.10	SQLite 3,40,0

Note

App Runner fournit un processus de compilation révisé pour des environnements d'exécution majeurs spécifiques qui ont été publiés plus récemment. Pour cette raison, vous verrez des références à la version révisée d'App Runner et à la version originale d'App Runner dans certaines sections de ce document. Pour plus d'informations, consultez Versions d'exécution gérées et build d'App Runner.

Utilisation de la plateforme Node.js

Important

App Runner mettra fin à la prise en charge de Node.js 12, Node.js 14, Node.js 16 et Node.js 18 le 1er décembre 2025. Pour des recommandations et de plus amples informations, consultezthe section called “Fin du support pour les versions d'exécution gérées”.

La plateforme AWS App Runner Node.js fournit des environnements d'exécution gérés. Chaque environnement d'exécution facilite la création et l'exécution de conteneurs avec des applications Web basées sur une version de Node.js. Lorsque vous utilisez un environnement d'exécution Node.js, App Runner démarre avec une image d'exécution Node.js gérée. Cette image est basée sur l'image Docker d'Amazon Linux et contient le package d'exécution pour une version de Node.js ainsi que certains outils. App Runner utilise cette image d'exécution gérée comme image de base et ajoute le code de votre application pour créer une image Docker. Il déploie ensuite cette image pour exécuter votre service Web dans un conteneur.

Vous spécifiez un environnement d'exécution pour votre service App Runner lorsque vous créez un service à l'aide de la console App Runner ou de l'opération CreateServiceAPI. Vous pouvez également spécifier un environnement d'exécution dans le cadre de votre code source. Utilisez le `runtime` mot clé dans un fichier de configuration App Runner que vous incluez dans votre référentiel de code. La convention de dénomination d'un environnement d'exécution géré est*<language-name><major-version>*.

Pour connaître les noms et versions d'exécution de Node.js valides, consultezthe section called “Informations sur la publication”.

App Runner met à jour le moteur d'exécution de votre service avec la dernière version à chaque déploiement ou mise à jour de service. Si votre application nécessite une version spécifique d'un environnement d'exécution géré, vous pouvez le spécifier à l'aide du `runtime-version` mot clé dans le fichier de configuration d'App Runner. Vous pouvez verrouiller n'importe quel niveau de version, y compris une version majeure ou mineure. App Runner apporte uniquement des mises à jour de niveau inférieur à l'environnement d'exécution de votre service.

Syntaxe de version pour les environnements d'exécution de Node.js : *`major`*`[.`*`minor`*`[.`*`patch`*`]]`

Par exemple : `22.14.0`

Les exemples suivants illustrent le verrouillage des versions :

- `22.14`— Verrouillez les versions majeures et mineures. App Runner met à jour uniquement les versions de correctif.
- `22.14.0`— Verrouillez vers une version de correctif spécifique. App Runner ne met pas à jour votre version d'exécution.

Rubriques

- Configuration d'exécution de Node.js

- Des légendes pour des versions d'exécution spécifiques
- Exemples d'exécution de Node.js
- Informations sur la version d'exécution de Node.js

Configuration d'exécution de Node.js

Lorsque vous choisissez un environnement d'exécution géré, vous devez également configurer, au minimum, créer et exécuter des commandes. Vous les configurez lors de la création ou de la mise à jour de votre service App Runner. Pour ce faire, vous pouvez utiliser l'une des méthodes suivantes :

- Utilisation de la console App Runner : spécifiez les commandes dans la section Configurer le build de l'onglet Processus de création ou configuration.
- Utilisation de l'API App Runner : appelez l'opération CreateServiceou UpdateServiceAPI. Spécifiez les commandes à l'aide des `StartCommand` membres `BuildCommand` et du type de CodeConfigurationValuesdonnées.
- Utilisation d'un fichier de configuration : spécifiez une ou plusieurs commandes de génération en trois phases de génération au maximum, ainsi qu'une seule commande d'exécution servant à démarrer votre application. Il existe d'autres paramètres de configuration facultatifs.

La fourniture d'un fichier de configuration est facultative. Lorsque vous créez un service App Runner à l'aide de la console ou de l'API, vous spécifiez si App Runner obtient vos paramètres de configuration directement lors de sa création ou à partir d'un fichier de configuration.

En ce qui concerne les environnements d'exécution de Node.js en particulier, vous pouvez également configurer le build et le runtime à l'aide d'un fichier JSON nommé `package.json` à la racine de votre référentiel source. À l'aide de ce fichier, vous pouvez configurer la version du moteur Node.js, les packages de dépendances et diverses commandes (applications en ligne de commande). Les gestionnaires de packages tels que npm ou yarn interprètent ce fichier comme entrée pour leurs commandes.

Par exemple :

- npm installinstalle les packages définis par le `devDependencies` nœud `dependencies` et dans`package.json`.
- npm startou npm run start exécute la commande définie par le `scripts/start` nœud dans`package.json`.

Voici un exemple de fichier `package.json`.

package.json

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

Pour plus d'informations`package.json`, voir Création d'un fichier package.json sur le site Web de npm Docs.

Conseils

- Si votre `package.json` fichier définit une start commande, vous pouvez l'utiliser comme run commande dans votre fichier de configuration App Runner, comme le montre l'exemple suivant.

  Example

  package.json

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

```
  }
}
```

apprunner.yaml

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

- Lorsque vous exécutez npm install dans votre environnement de développement, npm crée le fichier`package-lock.json`. Ce fichier contient un instantané des versions de package que npm vient d'installer. Par la suite, lorsque npm installe des dépendances, il utilise ces versions exactes. Si vous installez le fil, il crée un `yarn.lock` fichier. Enregistrez ces fichiers dans votre référentiel de code source pour vous assurer que votre application est installée avec les versions des dépendances avec lesquelles vous l'avez développée et testée.
- Vous pouvez également utiliser un fichier de configuration App Runner pour configurer la version Node.js et la commande de démarrage. Dans ce cas, ces définitions remplacent celles figurant dans`package.json`. Un conflit entre la node version `package.json` et la `runtime-version` valeur du fichier de configuration d'App Runner entraîne l'échec de la phase de création d'App Runner.

Des légendes pour des versions d'exécution spécifiques

Node.js 22 et Node.js 18 (version révisée d'App Runner)

App Runner exécute désormais un processus de génération mis à jour pour les applications basé sur les versions d'exécution suivantes : Python 3.11, Node.js 22 et Node.js 18. Si votre application s'exécute sur l'une de ces versions d'exécution, consultez Versions d'exécution gérées et build d'App Runner pour plus d'informations sur le processus de génération révisé. Les applications qui utilisent toutes les autres versions d'exécution ne sont pas affectées et continuent à utiliser le processus de génération d'origine.

Exemples d'exécution de Node.js

Les exemples suivants montrent les fichiers de configuration App Runner permettant de créer et d'exécuter un service Node.js.

> **Note**
>
> La version d'exécution utilisée dans ces exemples est`22.14.0`. Vous pouvez le remplacer par la version que vous souhaitez utiliser. Pour connaître la dernière version d'exécution de Node.js prise en charge, consultezthe section called “Informations sur la publication”.

Fichier de configuration Node.js minimal

Cet exemple montre un fichier de configuration minimale que vous pouvez utiliser avec un environnement d'exécution géré par Node.js. Pour les hypothèses formulées par App Runner avec un fichier de configuration minimal, consultezthe section called “Exemples de fichiers de configuration”.

Example apprunner.yaml

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

Fichier de configuration Node.js étendu

Cet exemple montre l'utilisation de toutes les clés de configuration avec un environnement d'exécution géré par Node.js.

> **Note**
>
> La version d'exécution utilisée dans ces exemples est`22.14.0`. Vous pouvez le remplacer par la version que vous souhaitez utiliser. Pour connaître la dernière version d'exécution de Node.js prise en charge, consultezthe section called “Informations sur la publication”.

Example apprunner.yaml

```
version: 1.0
runtime: nodejs22
build:
```

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

Fichier de configuration Node.js étendu — Node.js 22 (utilise la version révisée)

Cet exemple montre l'utilisation de toutes les clés de configuration avec un environnement d'exécution géré par Node.js dans le`apprunner.yaml`. Cet exemple inclut une `pre-run` section, car cette version de Node.js utilise la version révisée d'App Runner.

Le `pre-run` paramètre n'est pris en charge que par la version révisée d'App Runner. N'insérez pas ce paramètre dans votre fichier de configuration si votre application utilise des versions d'exécution prises en charge par la version originale d'App Runner. Pour de plus amples informations, veuillez consulter Versions d'exécution gérées et build d'App Runner.

> **Note**
>
> La version d'exécution utilisée dans ces exemples est`22.14.0`. Vous pouvez le remplacer par la version que vous souhaitez utiliser. Pour connaître la dernière version d'exécution de Node.js prise en charge, consultezthe section called “Informations sur la publication”.

Example apprunner.yaml

```
version: 1.0
```

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

Application Node.js avec Grunt

Cet exemple montre comment configurer une application Node.js développée avec Grunt. Grunt est un lanceur de JavaScript tâches en ligne de commande. Il exécute des tâches répétitives et gère l'automatisation des processus afin de réduire les erreurs humaines. Les plugins Grunt et Grunt sont installés et gérés à l'aide de npm. Vous configurez Grunt en incluant le `Gruntfile.js` fichier à la racine de votre dépôt source.

Example package.json

```
{
  "scripts": {
    "build": "grunt uglify",
    "start": "node app.js"
  },
  "devDependencies": {
    "grunt": "~0.4.5",
    "grunt-contrib-jshint": "~0.10.0",
```

```
    "grunt-contrib-nodeunit": "~0.4.1",
    "grunt-contrib-uglify": "~0.5.0"
  },
  "dependencies": {
    "express": "^4.15.2"
  },
}
```

Example Gruntfile.js

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

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

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

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

};
```

Example apprunner.yaml

> **Note**
>
> La version d'exécution utilisée dans ces exemples est*22.14.0*. Vous pouvez le remplacer par la version que vous souhaitez utiliser. Pour connaître la dernière version d'exécution de Node.js prise en charge, consultezthe section called “Informations sur la publication”.

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

Informations sur la version d'exécution de Node.js

> **Important**
>
> App Runner mettra fin à la prise en charge de Node.js 12, Node.js 14, Node.js 16 et Node.js 18 le 1er décembre 2025. Pour des recommandations et de plus amples informations, consultezthe section called “Fin du support pour les versions d'exécution gérées”.

> **Note**
>
> La politique d'obsolescence standard d'App Runner consiste à déprécier un environnement d'exécution lorsqu'un composant majeur de celui-ci atteint la fin du support communautaire à long terme (LTS) et que les mises à jour de sécurité ne sont plus disponibles. Dans certains cas, App Runner peut retarder la dépréciation d'un environnement d'exécution pendant une période limitée, au-delà de la end-of-support date de publication de la version linguistique prise en charge par le moteur d'exécution. Un exemple d'un tel cas pourrait être l'extension du support pour un environnement d'exécution afin de laisser aux clients le temps de migrer.

Cette rubrique répertorie tous les détails des versions d'exécution de Node.js prises en charge par App Runner.

Versions d'exécution prises en charge — version révisée d'App Runner

Nom de l’exécution	Versions mineures	Forfaits inclus
Node.js 22 (nodejs22)	22.21.1	npm 10.9.4, fil 1.22.22
	22,20,0	npm 10.9.3, fil 1.22.22
	22,17,0	npm 10.9.2, fil 1.22.22
	22,16,0	npm 10.9.2, fil 1.22.22
	22,14,0	npm 10.9.2, fil 1.22.22

> **Note**
>
> App Runner fournit un processus de compilation révisé pour des environnements d'exécution majeurs spécifiques qui ont été publiés plus récemment. Pour cette raison, vous verrez des références à la version révisée d'App Runner et à la version originale d'App Runner dans certaines sections de ce document. Pour plus d'informations, consultez Versions d'exécution gérées et build d'App Runner.

Versions d'exécution prises en charge — version révisée d'App Runner

Nom de l’exécution	Versions mineures	Forfaits inclus
Node.js 18 (nodejs18)	18,20,8	npm 10.8.2, fil 1.22.22
	18,20,7	npm 10.8.2, fil 1.22.22
	18,20,6	npm 10.8.2, fil 1.22.22
	18,20,5	npm 10.8.2, fil 1.22.22
	18,20,4	npm 10.7.0, fil 1.22.22
	18,20,3	npm 10.7.0, fil 1.22.22
	18,20,2	npm 10, fil *

Nom de l’exécution	Versions mineures	Forfaits inclus
	18,19.1	npm 10, fil *
	18,19,0	npm 10, fil *

Versions d'exécution prises en charge : version originale d'App Runner

Nom de l’exécution	Versions mineures	Forfaits inclus
Node.js 16 (nodejs16)	16,20,2	npm 8,19,4, fil 1,22,22
	16,20.1	npm 8.19.4, fil *
	16,20,0	npm 8.19.4, fil *
	16,19.1	npm 8.19.4, fil *
	16,19,0	npm 8.19.4, fil *
	16,18.1	npm 8.19.4, fil *
	16,17.1	npm 8.19.4, fil *
	16,17,0	npm 8.19.4, fil *
Node.js 14 (nodejs14)	14,21.3	npm 6.14,18, fil 1.22.22
	14.21.2	npm 6.14.18, fil *
	14.21.1	npm 6.14.18, fil *
	14.20.1	npm 6.14.18, fil *
	14,19,0	npm 6.14.18, fil *
Node.js 12 (nodejs12)	12,22.12	npm 6.14,16, fil 1.22.22
	12,21,0	npm 6.14.16, fil *

Utilisation de la plateforme Java

La plate-forme AWS App Runner Java fournit des environnements d'exécution gérés. Chaque environnement d'exécution facilite la création et l'exécution de conteneurs avec des applications Web basées sur une version Java. Lorsque vous utilisez un environnement d'exécution Java, App Runner démarre avec une image d'exécution Java gérée. Cette image est basée sur l'image Docker d'Amazon Linux et contient le package d'exécution pour une version de Java et certains outils. App Runner utilise cette image d'exécution gérée comme image de base et ajoute le code de votre application pour créer une image Docker. Il déploie ensuite cette image pour exécuter votre service Web dans un conteneur.

Vous spécifiez un environnement d'exécution pour votre service App Runner lorsque vous créez un service à l'aide de la console App Runner ou de l'opération CreateServiceAPI. Vous pouvez également spécifier un environnement d'exécution dans le cadre de votre code source. Utilisez le `runtime` mot clé dans un fichier de configuration App Runner que vous incluez dans votre référentiel de code. La convention de dénomination d'un environnement d'exécution géré est*<language-name><major-version>*.

À l'heure actuelle, tous les environnements d'exécution Java pris en charge sont basés sur Amazon Corretto. Pour les noms et versions d'exécution Java valides, consultezthe section called “Informations sur la publication”.

App Runner met à jour le moteur d'exécution de votre service avec la dernière version à chaque déploiement ou mise à jour de service. Si votre application nécessite une version spécifique d'un environnement d'exécution géré, vous pouvez le spécifier à l'aide du `runtime-version` mot clé dans le fichier de configuration d'App Runner. Vous pouvez verrouiller n'importe quel niveau de version, y compris une version majeure ou mineure. App Runner apporte uniquement des mises à jour de niveau inférieur à l'environnement d'exécution de votre service.

Syntaxe de version pour les environnements d'exécution Amazon Corretto :

Exécution	Syntaxe	Exemple
corretto11	`11.0[.`*`openjdk-update`* `[.`*`openjdk-build`* `[.`*`corretto-specific-revision`* `]]]`	`11.0.13.08.1`

Exécution	Syntaxe	Exemple
corretto8	`8[.`*`openjdk-update`* `[.`*`openjdk-build`* `[.`*`corretto-specific-revision`* `]]]`	`8.312.07.1`

Les exemples suivants illustrent le verrouillage des versions :

- `11.0.13`— Verrouillez la version de mise à jour d'Open JDK. App Runner met à jour uniquement les versions de niveau inférieur d'Open JDK et d'Amazon Corretto.
- `11.0.13.08.1`— Verrouillez vers une version spécifique. App Runner ne met pas à jour votre version d'exécution.

Rubriques

- Configuration de l'environnement d'exécution Java
- Exemples d'exécution Java
- Informations sur la version de Java Runtime

Configuration de l'environnement d'exécution Java

Lorsque vous choisissez un environnement d'exécution géré, vous devez également configurer, au minimum, créer et exécuter des commandes. Vous les configurez lors de la création ou de la mise à jour de votre service App Runner. Pour ce faire, vous pouvez utiliser l'une des méthodes suivantes :

- Utilisation de la console App Runner : spécifiez les commandes dans la section Configurer le build de l'onglet Processus de création ou configuration.
- Utilisation de l'API App Runner : appelez l'opération CreateServiceou UpdateServiceAPI. Spécifiez les commandes à l'aide des `StartCommand` membres `BuildCommand` et du type de CodeConfigurationValuesdonnées.
- Utilisation d'un fichier de configuration : spécifiez une ou plusieurs commandes de génération en trois phases de génération au maximum, ainsi qu'une seule commande d'exécution servant à démarrer votre application. Il existe d'autres paramètres de configuration facultatifs.

La fourniture d'un fichier de configuration est facultative. Lorsque vous créez un service App Runner à l'aide de la console ou de l'API, vous spécifiez si App Runner obtient vos paramètres de configuration directement lors de sa création ou à partir d'un fichier de configuration.

Exemples d'exécution Java

Les exemples suivants présentent les fichiers de configuration App Runner permettant de créer et d'exécuter un service Java. Le dernier exemple est le code source d'une application Java complète que vous pouvez déployer sur un service d'exécution Corretto 11.

> **Note**
>
> La version d'exécution utilisée dans ces exemples est`11.0.13.08.1`. Vous pouvez le remplacer par la version que vous souhaitez utiliser. Pour la dernière version d'exécution Java prise en charge, voirthe section called “Informations sur la publication”.

Fichier de configuration Minimal Corretto 11

Cet exemple montre un fichier de configuration minimal que vous pouvez utiliser avec un environnement d'exécution géré par Corretto 11. Pour les hypothèses formulées par App Runner avec un fichier de configuration minimal, consultez.

Example apprunner.yaml

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

Fichier de configuration étendu de Corretto 11

Cet exemple montre comment utiliser toutes les clés de configuration avec un environnement d'exécution géré par Corretto 11.

> **Note**
>
> La version d'exécution utilisée dans ces exemples est*11.0.13.08.1*. Vous pouvez le remplacer par la version que vous souhaitez utiliser. Pour la dernière version d'exécution Java prise en charge, voirthe section called “Informations sur la publication”.

Example apprunner.yaml

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

Source complète de l'application Corretto 11

Cet exemple montre le code source d'une application Java complète que vous pouvez déployer sur un service d'exécution Corretto 11.

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

```
package com.HelloWorld;

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

@RestController
public class HelloWorld {

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

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

```
package com.HelloWorld;

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

@SpringBootApplication
public class Main {

    public static void main(String[] args) {

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

Example apprunner.yaml

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

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

Example pom.xml

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

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

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

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

```
  </dependencies>

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

Informations sur la version de Java Runtime

Cette rubrique répertorie tous les détails des versions d'exécution Java prises en charge par App Runner.

Versions d'exécution prises en charge : version originale d'App Runner

Nom de l’exécution	Versions mineures	Forfaits inclus
Corretto 11 (Corretto 11)	11,0.28,6.1	Maven 3.9.11, Gradle 6.9.4
	11,0.27,6.1	Maven 3.9.10, Gradle 6.9.4
	11,0.27,6.1	Maven 3.9.9, Gradle 6.9.4
	11,0.26,4.1	Maven 3.9.9, Gradle 6.9.4
	11,0.25,9.1	Maven 3.9.9, Gradle 6.9.4
	11,0.24.8.1	Maven 3.9.9, Gradle 6.9.4
	11,0.23,9.1	Maven 3.9.8, Gradle 6.9.4

Nom de l’exécution	Versions mineures	Forfaits inclus
	11,0.22.7.1	Maven 3.9.6, Gradle 6.9.4
	11,0.21,9.1	Maven 3.9.6, Gradle 6.9.4
	11,0.21,9.1	Maven 3.9.5, Gradle 6.9.4
	11,0.20,8.1	Maven 3.9.3, Gradle 6.9.4
	11,0.19,7.1	Maven 3.9.3, Gradle 6.9.4
	11,0.18.10.1	Maven 3.9.1, Gradle 6.9.4
	11,0.17,8.1	Maven 3.8.6, Gradle 6.9.3
	11,0.16,9.1	Maven 3.8.6, Gradle 6.9.2
	11,0.13,08.1	Maven 3.6.3, Gradle 6.5
Corretto 8 (corretto 8)	8,472,08.1	Maven 3.9.11, Gradle 6.9.4
	8,462,08.1	Maven 3.9.11, Gradle 6.9.4
	8,452,09,2	Maven 3.9.10, Gradle 6.9.4
	8,452,09,2	Maven 3.9.9, Gradle 6.9.4
	8,452,09.1	Maven 3.9.9, Gradle 6.9.4
	8,442,06.1	Maven 3.9.9, Gradle 6.9.4
	8,432,06.1	Maven 3.9.9, Gradle 6.9.4
	8,422,05.1	Maven 3.9.9, Gradle 6.9.4
	8,412,08,1	Maven 3.9.8, Gradle 6.9.4
	8,402,08.1	Maven 3.9.6, Gradle 6.9.4
	8,392,08,1	Maven 3.9.6, Gradle 6.9.4

Nom de l'exécution	Versions mineures	Forfaits inclus
	8,382,05.1	Maven 3.9.4, Gradle 6.9.4
	8,372,07.1	Maven 3.9.3, Gradle 6.9.4
	8,362,08.1	Maven 3.9.1, Gradle 6.9.4
	8,352,08.1	Maven 3.8.6, Gradle 6.9.3
	8,342,07.4	Maven 3.8.6, Gradle 6.9.2
	8,312,07,1	Maven 3.6.3, Gradle 6.5

> **Note**
>
> App Runner fournit un processus de compilation révisé pour des environnements d'exécution majeurs spécifiques qui ont été publiés plus récemment. Pour cette raison, vous verrez des références à la version révisée d'App Runner et à la version originale d'App Runner dans certaines sections de ce document. Pour plus d'informations, consultez Versions d'exécution gérées et build d'App Runner.

Utilisation de la plateforme .NET

> **Important**
>
> App Runner mettra fin au support de .NET 6 le 1er décembre 2025. Pour des recommandations et de plus amples informations, consultezthe section called "Fin du support pour les versions d'exécution gérées".

La plate-forme AWS App Runner .NET fournit des environnements d'exécution gérés. Chaque environnement d'exécution facilite la création et l'exécution de conteneurs avec des applications Web basées sur une version .NET. Lorsque vous utilisez un environnement d'exécution .NET, App Runner démarre avec une image d'exécution .NET gérée. Cette image est basée sur l'image Docker d'Amazon Linux et contient le package d'exécution d'une version de .NET ainsi que certains outils et packages de dépendances populaires. App Runner utilise cette image d'exécution gérée comme

image de base et ajoute le code de votre application pour créer une image Docker. Il déploie ensuite cette image pour exécuter votre service Web dans un conteneur.

Vous spécifiez un environnement d'exécution pour votre service App Runner lorsque vous créez un service à l'aide de la console App Runner ou de l'opération CreateServiceAPI. Vous pouvez également spécifier un environnement d'exécution dans le cadre de votre code source. Utilisez le `runtime` mot clé dans un fichier de configuration App Runner que vous incluez dans votre référentiel de code. La convention de dénomination d'un environnement d'exécution géré est*<language-name><major-version>*.

Pour connaître les noms et versions d'environnement d'exécution .NET valides, consultezthe section called “Informations sur la publication”.

App Runner met à jour le moteur d'exécution de votre service avec la dernière version à chaque déploiement ou mise à jour de service. Si votre application nécessite une version spécifique d'un environnement d'exécution géré, vous pouvez le spécifier à l'aide du `runtime-version` mot clé dans le fichier de configuration d'App Runner. Vous pouvez verrouiller n'importe quel niveau de version, y compris une version majeure ou mineure. App Runner apporte uniquement des mises à jour de niveau inférieur à l'environnement d'exécution de votre service.

Syntaxe de version pour les environnements d'exécution .NET : *major*`[.`*minor*`[.`*patch*`]]`

Par exemple : `6.0.9`

Les exemples suivants illustrent le verrouillage des versions :

- `6.0`— Verrouillez les versions majeures et mineures. App Runner met à jour uniquement les versions de correctif.
- `6.0.9`— Verrouillez vers une version de correctif spécifique. App Runner ne met pas à jour votre version d'exécution.

Rubriques

- Configuration de l'environnement d'exécution .NET
- Exemples d'exécution .NET
- Informations sur la version de .NET Runtime

Configuration de l'environnement d'exécution .NET

Lorsque vous choisissez un environnement d'exécution géré, vous devez également configurer, au minimum, créer et exécuter des commandes. Vous les configurez lors de la création ou de la mise à jour de votre service App Runner. Pour ce faire, vous pouvez utiliser l'une des méthodes suivantes :

- Utilisation de la console App Runner : spécifiez les commandes dans la section Configurer le build de l'onglet Processus de création ou configuration.
- Utilisation de l'API App Runner : appelez l'opération CreateServiceou UpdateServiceAPI. Spécifiez les commandes à l'aide des `StartCommand` membres `BuildCommand` et du type de CodeConfigurationValuesdonnées.
- Utilisation d'un fichier de configuration : spécifiez une ou plusieurs commandes de génération en trois phases de génération au maximum, ainsi qu'une seule commande d'exécution servant à démarrer votre application. Il existe d'autres paramètres de configuration facultatifs.

La fourniture d'un fichier de configuration est facultative. Lorsque vous créez un service App Runner à l'aide de la console ou de l'API, vous spécifiez si App Runner obtient vos paramètres de configuration directement lors de sa création ou à partir d'un fichier de configuration.

Exemples d'exécution .NET

Les exemples suivants présentent les fichiers de configuration App Runner permettant de créer et d'exécuter un service .NET. Le dernier exemple est le code source d'une application .NET complète que vous pouvez déployer sur un service d'exécution .NET.

> **Note**
>
> La version d'exécution utilisée dans ces exemples est`6.0.9`. Vous pouvez le remplacer par la version que vous souhaitez utiliser. Pour connaître la dernière version d'exécution .NET prise en charge, consultezthe section called “Informations sur la publication”.

Fichier de configuration .NET minimal

Cet exemple montre un fichier de configuration minimal que vous pouvez utiliser avec un environnement d'exécution géré .NET. Pour les hypothèses formulées par App Runner avec un fichier de configuration minimal, consultezthe section called “Exemples de fichiers de configuration”.

Example apprunner.yaml

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

Fichier de configuration .NET étendu

Cet exemple montre l'utilisation de toutes les clés de configuration avec un environnement d'exécution géré par .NET.

> **Note**
>
> La version d'exécution utilisée dans ces exemples est*6.0.9*. Vous pouvez le remplacer par la version que vous souhaitez utiliser. Pour connaître la dernière version d'exécution .NET prise en charge, consultezthe section called “Informations sur la publication”.

Example apprunner.yaml

```
version: 1.0
runtime: dotnet6
build:
  commands:
    pre-build:
      - scripts/prebuild.sh
    build:
      - dotnet publish -c Release -o out
    post-build:
      - scripts/postbuild.sh
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
run:
  runtime-version: 6.0.9
  command: dotnet out/HelloWorldDotNetApp.dll
```

```
  network:
    port: 5000
    env: APP_PORT
  env:
    - name: ASPNETCORE_URLS
      value: "http://*:5000"
```

Source d'application .NET complète

Cet exemple montre le code source d'une application .NET complète que vous pouvez déployer sur un service d'exécution .NET.

> **Note**
>
> - Exécutez la commande suivante pour créer une application Web .NET 6 simple : `dotnet new web --name HelloWorldDotNetApp -f net6.0`
> - Ajoutez le `apprunner.yaml` à l'application Web .NET 6 créée.

Example HelloWorldDotNetApp

```
version: 1.0
runtime: dotnet6
build:
  commands:
    build:
      - dotnet publish -c Release -o out
run:
  command: dotnet out/HelloWorldDotNetApp.dll
  network:
    port: 5000
    env: APP_PORT
  env:
    - name: ASPNETCORE_URLS
      value: "http://*:5000"
```

Informations sur la version de .NET Runtime

> **Important**
>
> App Runner mettra fin au support de .NET 6 le 1er décembre 2025. Pour des recommandations et de plus amples informations, consultezthe section called “Fin du support pour les versions d'exécution gérées”.

Cette rubrique fournit des informations complètes sur les versions d'exécution .NET prises en charge par App Runner.

Versions d'exécution prises en charge : version originale d'App Runner

Nom de l’exécution	Versions mineures	Forfaits inclus
.NET 6 (dotnet6)	6,0,36	.NET SDK 6.0.428
	6,0,33	.NET SDK 6.0.425
	6,0,32	.NET SDK 6.0.424
	6,0,31	.NET SDK 6.0.423
	6,0,30	.NET SDK 6.0.422
	6,0,29	.NET SDK 6.0.421
	6,0,28	.NET SDK 6.0.420
	6,0,26	.NET SDK 6.0.418
	6,0,25	.NET SDK 6.0.417
	6,0,24	.NET SDK 6.0.416
	6,0,22	.NET SDK 6.0.414
	6,0,21	.NET SDK 6.0.413
	6,0,20	.NET SDK 6.0.412

Nom de l'exécution	Versions mineures	Forfaits inclus
	6,0,19	.NET SDK 6.0.411
	6,0,16	.NET SDK 6.0.408
	6,0,15	.NET SDK 6.0.407
	6,0,14	.NET SDK 6.0.406
	6,0,13	.NET SDK 6.0.405
	6,0,12	.NET SDK 6.0.404
	6,0,11	.NET SDK 6.0.403
	6,0.10	.NET SDK 6.0.402
	6,0,9	.NET SDK 6.0.401

Note

App Runner fournit un processus de compilation révisé pour des environnements d'exécution majeurs spécifiques qui ont été publiés plus récemment. Pour cette raison, vous verrez des références à la version révisée d'App Runner et à la version originale d'App Runner dans certaines sections de ce document. Pour plus d'informations, consultez Versions d'exécution gérées et build d'App Runner.

À l'aide de la plateforme PHP

Important

App Runner mettra fin au support de PHP 8.1 le 31 décembre 2025. Pour des recommandations et de plus amples informations, consultezthe section called "Fin du support pour les versions d'exécution gérées".

La plateforme AWS App Runner PHP fournit des environnements d'exécution gérés. Vous pouvez utiliser chaque environnement d'exécution pour créer et exécuter des conteneurs avec des applications Web basées sur une version de PHP. Lorsque vous utilisez un environnement d'exécution PHP, App Runner démarre avec une image d'exécution PHP gérée. Cette image est basée sur l'image Docker d'Amazon Linux et contient le package d'exécution pour une version de PHP et certains outils. App Runner utilise cette image d'exécution gérée comme image de base et ajoute le code de votre application pour créer une image Docker. Il déploie ensuite cette image pour exécuter votre service Web dans un conteneur.

Vous spécifiez un environnement d'exécution pour votre service App Runner lorsque vous créez un service à l'aide de la console App Runner ou de l'opération CreateServiceAPI. Vous pouvez également spécifier un environnement d'exécution dans le cadre de votre code source. Utilisez le `runtime` mot clé dans un fichier de configuration App Runner que vous incluez dans votre référentiel de code. La convention de dénomination d'un environnement d'exécution géré est*`<language-name><major-version>`*.

Pour les noms et versions d'exécution PHP valides, consultezthe section called “Informations sur la publication”.

App Runner met à jour le moteur d'exécution de votre service avec la dernière version à chaque déploiement ou mise à jour de service. Si votre application nécessite une version spécifique d'un environnement d'exécution géré, vous pouvez le spécifier à l'aide du `runtime-version` mot clé dans le fichier de configuration d'App Runner. Vous pouvez verrouiller n'importe quel niveau de version, y compris une version majeure ou mineure. App Runner apporte uniquement des mises à jour de niveau inférieur à l'environnement d'exécution de votre service.

Syntaxe de version pour les environnements d'exécution PHP : *`major`*`[.`*`minor`*`[.`*`patch`*`]]`

Par exemple : `8.1.10`

Voici des exemples de verrouillage de version :

- `8.1`— Verrouillez les versions majeures et mineures. App Runner met à jour uniquement les versions de correctif.
- `8.1.10`— Verrouillez vers une version de correctif spécifique. App Runner ne met pas à jour votre version d'exécution.

> **Important**
>
> Si vous souhaitez spécifier le répertoire source du référentiel de code pour votre service App Runner dans un emplacement autre que le répertoire racine du dépôt par défaut, votre version d'exécution gérée de PHP doit être PHP `8.1.22` ou une version ultérieure. Les versions d'exécution de PHP antérieures ne `8.1.22` peuvent utiliser que le répertoire source racine par défaut.

Rubriques

- Configuration de l'environnement d'exécution PHP
- Compatibilité
- Exemples d'exécution PHP
- Informations sur la version d'exécution de PHP

Configuration de l'environnement d'exécution PHP

Lorsque vous choisissez un environnement d'exécution géré, vous devez également configurer, au minimum, créer et exécuter des commandes. Vous les configurez lors de la création ou de la mise à jour de votre service App Runner. Pour ce faire, vous pouvez utiliser l'une des méthodes suivantes :

- Utilisation de la console App Runner : spécifiez les commandes dans la section Configurer le build de l'onglet Processus de création ou configuration.
- Utilisation de l'API App Runner : appelez l'opération CreateServiceou UpdateServiceAPI. Spécifiez les commandes à l'aide des `StartCommand` membres `BuildCommand` et du type de CodeConfigurationValuesdonnées.
- Utilisation d'un fichier de configuration : spécifiez une ou plusieurs commandes de génération en trois phases de génération au maximum, ainsi qu'une seule commande d'exécution servant à démarrer votre application. Il existe d'autres paramètres de configuration facultatifs.

La fourniture d'un fichier de configuration est facultative. Lorsque vous créez un service App Runner à l'aide de la console ou de l'API, vous spécifiez si App Runner obtient vos paramètres de configuration directement lors de sa création ou à partir d'un fichier de configuration.

Compatibilité

Vous pouvez exécuter vos services App Runner sur la plateforme PHP en utilisant l'un des serveurs Web suivants :

- Apache HTTP Server
- NGINX

Apache HTTP Serveret NGINX sont compatibles avec PHP-FPM. Vous pouvez démarrer Apache HTTP Serveret NGINXen utilisant l'une des méthodes suivantes :

- Supervisord - Pour plus d'informations sur l'exécution d'un superviseur supervisord, consultez Exécuter un superviseur.
- Script de démarrage

Pour des exemples sur la façon de configurer votre service App Runner avec la plate-forme PHP à l'aide du serveur HTTP Apache ou de NGINX, consultez. the section called “Source complète de l'application PHP”

Structure de fichier

`index.php`Il doit être installé dans le `public` dossier situé sous le `root` répertoire du serveur Web.

> **Note**
>
> Nous recommandons de stocker `supervisord.conf` les fichiers `startup.sh` or dans le répertoire racine du serveur Web. Assurez-vous que la `start` commande pointe vers l'emplacement où les `supervisord.conf` fichiers `startup.sh` ou sont stockés.

Voici un exemple de structure de fichier que vous utilisez supervisord.

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

Voici un exemple de structure de fichier si vous utilisez un script de démarrage.

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

Nous vous recommandons de stocker ces structures de fichiers dans le répertoire source du référentiel de code désigné pour le service App Runner.

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

> **Important**
>
> Si vous souhaitez spécifier le répertoire source du référentiel de code pour votre service App Runner dans un emplacement autre que le répertoire racine du dépôt par défaut, votre version d'exécution gérée de PHP doit être PHP `8.1.22` ou une version ultérieure. Les versions d'exécution de PHP antérieures ne `8.1.22` peuvent utiliser que le répertoire source racine par défaut.
>
> App Runner met à jour le moteur d'exécution de votre service avec la dernière version à chaque déploiement ou mise à jour de service. Votre service utilisera les environnements d'exécution les plus récents par défaut, sauf si vous avez spécifié le verrouillage de version à l'aide du `runtime-version` mot clé dans le fichier de configuration d'App Runner.

Exemples d'exécution PHP

Voici des exemples de fichiers de configuration App Runner utilisés pour créer et exécuter un service PHP.

Fichier de configuration PHP minimal

L'exemple suivant est un fichier de configuration minimale que vous pouvez utiliser avec un environnement d'exécution géré en PHP. Pour plus d'informations sur un fichier de configuration minimale, consultezthe section called “Exemples de fichiers de configuration”.

Example apprunner.yaml

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

Fichier de configuration PHP étendu

L'exemple suivant utilise toutes les clés de configuration avec un environnement d'exécution géré par PHP.

> **Note**
>
> La version d'exécution utilisée dans ces exemples est*8.1.10*. Vous pouvez le remplacer par la version que vous souhaitez utiliser. Pour la dernière version d'exécution de PHP prise en charge, voirthe section called “Informations sur la publication”.

Example apprunner.yaml

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

```
  network:
    port: 5000
    env: APP_PORT
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
```

Source complète de l'application PHP

Les exemples suivants concernent le code source d'une application PHP que vous pouvez utiliser pour déployer sur un service d'exécution PHP à l'aide de Apache HTTP Serverou NGINX. Ces exemples supposent que vous utilisez la structure de fichier par défaut.

Exécution de la plate-forme PHP en Apache HTTP Server utilisant supervisord

Example Structure de fichier

> **Note**
>
> - Le `supervisord.conf` fichier peut être stocké n'importe où dans le référentiel. Assurez-vous que la `start` commande pointe vers l'endroit où le `supervisord.conf` fichier est stocké.
> - `index.php`Il doit être installé dans le `public` dossier situé sous le `root` répertoire.

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

Example supervisord.conf

```
[supervisord]
nodaemon=true

[program:httpd]
command=httpd -DFOREGROUND
autostart=true
```

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

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

Example apprunner.yaml

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

Example index.php

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

Exécution de la plate-forme PHP en Apache HTTP Server utilisant startup script

Example Structure de fichier

> **Note**
>
> - Le `startup.sh` fichier peut être stocké n'importe où dans le référentiel. Assurez-vous que la `start` commande pointe vers l'endroit où le `startup.sh` fichier est stocké.
> - `index.php`Il doit être installé dans le `public` dossier situé sous le `root` répertoire.

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

Example startup.sh

```
#!/bin/bash

set -o monitor

trap exit SIGCHLD

# Start apache
httpd -DFOREGROUND &

# Start php-fpm
php-fpm -F &

wait
```

> **Note**
>
> - Assurez-vous d'enregistrer le `startup.sh` fichier en tant qu'exécutable avant de le valider dans un dépôt Git. `chmod +x startup.sh`À utiliser pour définir l'autorisation d'exécution sur votre `startup.sh` fichier.

- Si vous n'enregistrez pas le `startup.sh` fichier en tant qu'exécutable, entrez `chmod +x startup.sh` la `build` commande dans votre `apprunner.yaml` fichier.

Example apprunner.yaml

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

Example index.php

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

Exécution de la plate-forme PHP en NGINX utilisant supervisord

Example Structure de fichier

> **Note**
>
> - Le `supervisord.conf` fichier peut être stocké n'importe où dans le référentiel. Assurez-vous que la `start` commande pointe vers l'endroit où le `supervisord.conf` fichier est stocké.

- `index.php`Il doit être installé dans le `public` dossier situé sous le `root` répertoire.

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

Example supervisord.conf

```
[supervisord]
nodaemon=true

[program:nginx]
command=nginx -g "daemon off;"
autostart=true
autorestart=true
stdout_logfile=/dev/stdout
stdout_logfile_maxbytes=0
stderr_logfile=/dev/stderr
stderr_logfile_maxbytes=0

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

Example apprunner.yaml

```
version: 1.0
runtime: php81
build:
  commands:
    build:
      - PYTHON=python2 amazon-linux-extras install epel
      - yum -y install supervisor
```

```
run:
  command: supervisord
  network:
    port: 8080
    env: APP_PORT
```

Example index.php

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

Exécution de la plate-forme PHP en NGINX utilisant startup script

Example Structure de fichier

> **Note**
>
> - Le `startup.sh` fichier peut être stocké n'importe où dans le référentiel. Assurez-vous que la `start` commande pointe vers l'endroit où le `startup.sh` fichier est stocké.
> - `index.php`Il doit être installé dans le `public` dossier situé sous le `root` répertoire.

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

Example startup.sh

```
#!/bin/bash

set -o monitor
```

```
trap exit SIGCHLD

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

# Start php-fpm
php-fpm -F &

wait
```

> **Note**
>
> - Assurez-vous d'enregistrer le `startup.sh` fichier en tant qu'exécutable avant de le valider dans un dépôt Git. `chmod +x startup.sh`À utiliser pour définir l'autorisation d'exécution sur votre `startup.sh` fichier.
> - Si vous n'enregistrez pas le `startup.sh` fichier en tant qu'exécutable, entrez `chmod +x startup.sh` la `build` commande dans votre `apprunner.yaml` fichier.

Example apprunner.yaml

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

Example index.php

```
<html>
<head> <title>First PHP App</title> </head>
<body>
<?php
```

```
    print("Hello World!");
    print("<br>");
?>
</body>
</html>
```

Informations sur la version d'exécution de PHP

> **Important**
>
> App Runner mettra fin au support de PHP 8.1 le 31 décembre 2025. Pour des recommandations et de plus amples informations, consultezthe section called “Fin du support pour les versions d'exécution gérées”.

Cette rubrique répertorie tous les détails des versions d'exécution de PHP prises en charge par App Runner.

Versions d'exécution prises en charge : version originale d'App Runner

Nom de l’exécution	Versions mineures	Forfaits inclus
PHP 8.1 (php81)	8,133	
	8,1,32	
	8,1,31	
	8,1,29	
	8,128	
	8,1,27	
	8,1,26	
	8,124	
	8,122	
	8,1.21	

Nom de l'exécution	Versions mineures	Forfaits inclus
	8,1,20	
	8,19	
	8,117	
	8,116	
	8,114	
	8,113	
	8,12	
	8,110	

> **Note**
>
> App Runner fournit un processus de compilation révisé pour des environnements d'exécution majeurs spécifiques qui ont été publiés plus récemment. Pour cette raison, vous verrez des références à la version révisée d'App Runner et à la version originale d'App Runner dans certaines sections de ce document. Pour plus d'informations, consultez Versions d'exécution gérées et build d'App Runner.

Utilisation de la plateforme Ruby

> **Important**
>
> App Runner mettra fin au support de Ruby 3.1 le 1er décembre 2025. Pour des recommandations et de plus amples informations, consultezthe section called "Fin du support pour les versions d'exécution gérées".

La plateforme AWS App Runner Ruby fournit des environnements d'exécution gérés. Chaque environnement d'exécution facilite la création et l'exécution de conteneurs avec des applications

Web basées sur une version Ruby. Lorsque vous utilisez un environnement d'exécution Ruby, App Runner démarre avec une image d'exécution Ruby gérée. Cette image est basée sur l'image Docker d'Amazon Linux et contient le package d'exécution pour une version de Ruby et certains outils. App Runner utilise cette image d'exécution gérée comme image de base et ajoute le code de votre application pour créer une image Docker. Il déploie ensuite cette image pour exécuter votre service Web dans un conteneur.

Vous spécifiez un environnement d'exécution pour votre service App Runner lorsque vous créez un service à l'aide de la console App Runner ou de l'opération CreateServiceAPI. Vous pouvez également spécifier un environnement d'exécution dans le cadre de votre code source. Utilisez le `runtime` mot clé dans un fichier de configuration App Runner que vous incluez dans votre référentiel de code. La convention de dénomination d'un environnement d'exécution géré est*`<language-name><major-version>`*.

Pour les noms et versions d'exécution Ruby valides, consultezthe section called “Informations sur la publication”.

App Runner met à jour le moteur d'exécution de votre service avec la dernière version à chaque déploiement ou mise à jour de service. Si votre application nécessite une version spécifique d'un environnement d'exécution géré, vous pouvez le spécifier à l'aide du `runtime-version` mot clé dans le fichier de configuration d'App Runner. Vous pouvez verrouiller n'importe quel niveau de version, y compris une version majeure ou mineure. App Runner apporte uniquement des mises à jour de niveau inférieur à l'environnement d'exécution de votre service.

Syntaxe de version pour les environnements d'exécution Ruby : *`major`*`[.`*`minor`*`[.`*`patch`*`]]`

Par exemple : `3.1.2`

Les exemples suivants illustrent le verrouillage des versions :

- `3.1`— Verrouillez les versions majeures et mineures. App Runner met à jour uniquement les versions de correctif.
- `3.1.2`— Verrouillez vers une version de correctif spécifique. App Runner ne met pas à jour votre version d'exécution.

Rubriques

- Configuration du runtime Ruby
- Exemples d'exécution Ruby

- Informations sur la version de Ruby Runtime

Configuration du runtime Ruby

Lorsque vous choisissez un environnement d'exécution géré, vous devez également configurer, au minimum, créer et exécuter des commandes. Vous les configurez lors de la création ou de la mise à jour de votre service App Runner. Pour ce faire, vous pouvez utiliser l'une des méthodes suivantes :

- Utilisation de la console App Runner : spécifiez les commandes dans la section Configurer le build de l'onglet Processus de création ou configuration.
- Utilisation de l'API App Runner : appelez l'opération CreateServiceou UpdateServiceAPI. Spécifiez les commandes à l'aide des `StartCommand` membres `BuildCommand` et du type de CodeConfigurationValuesdonnées.
- Utilisation d'un fichier de configuration : spécifiez une ou plusieurs commandes de génération en trois phases de génération au maximum, ainsi qu'une seule commande d'exécution servant à démarrer votre application. Il existe d'autres paramètres de configuration facultatifs.

La fourniture d'un fichier de configuration est facultative. Lorsque vous créez un service App Runner à l'aide de la console ou de l'API, vous spécifiez si App Runner obtient vos paramètres de configuration directement lors de sa création ou à partir d'un fichier de configuration.

Exemples d'exécution Ruby

Les exemples suivants montrent les fichiers de configuration d'App Runner permettant de créer et d'exécuter un service Ruby.

Fichier de configuration Ruby minimal

Cet exemple montre un fichier de configuration minimal que vous pouvez utiliser avec un environnement d'exécution géré par Ruby. Pour les hypothèses formulées par App Runner avec un fichier de configuration minimal, consultezthe section called “Exemples de fichiers de configuration”.

Example apprunner.yaml

```
version: 1.0
runtime: ruby31
build:
  commands:
```

```
    build:
      - bundle install
run:
  command: bundle exec rackup --host 0.0.0.0 -p 8080
```

Fichier de configuration Ruby étendu

Cet exemple montre l'utilisation de toutes les clés de configuration avec un environnement d'exécution géré par Ruby.

> **Note**
>
> La version d'exécution utilisée dans ces exemples est*3.1.2*. Vous pouvez le remplacer par la version que vous souhaitez utiliser. Pour la dernière version d'exécution de Ruby prise en charge, voirthe section called “Informations sur la publication”.

Example apprunner.yaml

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

Source complète de l'application Ruby

Ces exemples montrent le code source d'une application Ruby complète que vous pouvez déployer sur un service d'exécution Ruby.

Example serveur.rb

```
# server.rb
require 'sinatra'

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

Example config.ru

```
# config.ru

require './server'

run Sinatra::Application
```

Example Gemfile

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

gem 'sinatra'
gem 'puma'
```

Example apprunner.yaml

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

```
    env: APP_PORT
```

Informations sur la version de Ruby Runtime

> **Important**
>
> App Runner mettra fin au support de Ruby 3.1 le 1er décembre 2025. Pour des recommandations et de plus amples informations, consultezthe section called “Fin du support pour les versions d'exécution gérées”.

Cette rubrique répertorie tous les détails des versions d'exécution de Ruby prises en charge par App Runner.

Versions d'exécution prises en charge : version originale d'App Runner

Nom de l’exécution	Versions mineures	Forfaits inclus
Ruby 3.1 (ruby31)	3.1.7	SQLite 3,50,2
	3.1.7	SQLite 3,50,1
	3.1.7	SQLite 3,50,0
	3.1.6	SQLite 3,49.1
	3.1.4	SQLite 3,46,0
	3.1.3	SQLite 3,41,0
	3.1.2	SQLite 3,39,4

> **Note**
>
> App Runner fournit un processus de compilation révisé pour des environnements d'exécution majeurs spécifiques qui ont été publiés plus récemment. Pour cette raison, vous verrez des références à la version révisée d'App Runner et à la version originale d'App Runner dans certaines sections de ce document. Pour plus d'informations, consultez Versions d'exécution gérées et build d'App Runner.

Utilisation de la plateforme Go

> **Important**
>
> App Runner mettra fin au support de Go 1.18 le 1er décembre 2025. Pour des recommandations et de plus amples informations, consultezthe section called “Fin du support pour les versions d'exécution gérées”.

La plateforme AWS App Runner Go fournit des environnements d'exécution gérés. Chaque environnement d'exécution facilite la création et l'exécution de conteneurs avec des applications Web basées sur une version Go. Lorsque vous utilisez un environnement d'exécution Go, App Runner démarre avec une image d'exécution Go gérée. Cette image est basée sur l'image Docker d'Amazon Linux et contient le package d'exécution pour une version de Go ainsi que certains outils. App Runner utilise cette image d'exécution gérée comme image de base et ajoute le code de votre application pour créer une image Docker. Il déploie ensuite cette image pour exécuter votre service Web dans un conteneur.

Vous spécifiez un environnement d'exécution pour votre service App Runner lorsque vous créez un service à l'aide de la console App Runner ou de l'opération CreateServiceAPI. Vous pouvez également spécifier un environnement d'exécution dans le cadre de votre code source. Utilisez le `runtime` mot clé dans un fichier de configuration App Runner que vous incluez dans votre référentiel de code. La convention de dénomination d'un environnement d'exécution géré est*`<language-name><major-version>`*.

Pour les noms et versions d'exécution Go valides, consultezthe section called “Informations sur la publication”.

App Runner met à jour le moteur d'exécution de votre service avec la dernière version à chaque déploiement ou mise à jour de service. Si votre application nécessite une version spécifique d'un environnement d'exécution géré, vous pouvez le spécifier à l'aide du `runtime-version` mot clé dans le fichier de configuration d'App Runner. Vous pouvez verrouiller n'importe quel niveau de version, y compris une version majeure ou mineure. App Runner apporte uniquement des mises à jour de niveau inférieur à l'environnement d'exécution de votre service.

Syntaxe de version pour les environnements d'exécution Go : *`major`*`[.`*`minor`*`[.`*`patch`*`]]`

Par exemple : `1.18.7`

Les exemples suivants illustrent le verrouillage des versions :

- `1.18`— Verrouillez les versions majeures et mineures. App Runner met à jour uniquement les versions de correctif.
- `1.18.7`— Verrouillez vers une version de correctif spécifique. App Runner ne met pas à jour votre version d'exécution.

Rubriques

- Configuration de l'environnement d'exécution Go
- Exemples d'exécution Go
- Informations sur les versions de Go Runtime

Configuration de l'environnement d'exécution Go

Lorsque vous choisissez un environnement d'exécution géré, vous devez également configurer, au minimum, créer et exécuter des commandes. Vous les configurez lors de la création ou de la mise à jour de votre service App Runner. Pour ce faire, vous pouvez utiliser l'une des méthodes suivantes :

- Utilisation de la console App Runner : spécifiez les commandes dans la section Configurer le build de l'onglet Processus de création ou configuration.
- Utilisation de l'API App Runner : appelez l'opération CreateServiceou UpdateServiceAPI. Spécifiez les commandes à l'aide des `StartCommand` membres `BuildCommand` et du type de CodeConfigurationValuesdonnées.
- Utilisation d'un fichier de configuration : spécifiez une ou plusieurs commandes de génération en trois phases de génération au maximum, ainsi qu'une seule commande d'exécution servant à démarrer votre application. Il existe d'autres paramètres de configuration facultatifs.

La fourniture d'un fichier de configuration est facultative. Lorsque vous créez un service App Runner à l'aide de la console ou de l'API, vous spécifiez si App Runner obtient vos paramètres de configuration directement lors de sa création ou à partir d'un fichier de configuration.

Exemples d'exécution Go

Les exemples suivants montrent les fichiers de configuration App Runner permettant de créer et d'exécuter un service Go.

Fichier de configuration Minimal Go

Cet exemple montre un fichier de configuration minimal que vous pouvez utiliser avec un environnement d'exécution géré par Go. Pour les hypothèses formulées par App Runner avec un fichier de configuration minimal, consultezthe section called “Exemples de fichiers de configuration”.

Example apprunner.yaml

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

Fichier de configuration Extended Go

Cet exemple montre l'utilisation de toutes les clés de configuration avec un environnement d'exécution géré par Go.

> **Note**
>
> La version d'exécution utilisée dans ces exemples est*1.18.7*. Vous pouvez le remplacer par la version que vous souhaitez utiliser. Pour la dernière version d'exécution Go prise en charge, voirthe section called “Informations sur la publication”.

Example apprunner.yaml

```
version: 1.0
runtime: go1
build:
  commands:
     pre-build:
      - scripts/prebuild.sh
    build:
      - go build main.go
    post-build:
      - scripts/postbuild.sh
```

```
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
run:
  runtime-version: 1.18.7
  command: ./main
  network:
    port: 3000
    env: APP_PORT
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
```

Source complète de l'application Go

Ces exemples montrent le code source d'une application Go complète que vous pouvez déployer sur un service d'exécution Go.

Example main.go

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

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

Example apprunner.yaml

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

```
  command: ./main
  network:
    port: 3000
    env: APP_PORT
```

Informations sur les versions de Go Runtime

> **Important**
>
> App Runner mettra fin au support de Go 1.18 le 1er décembre 2025. Pour des recommandations et de plus amples informations, consultezthe section called “Fin du support pour les versions d'exécution gérées”.

Cette rubrique répertorie tous les détails des versions d'exécution Go prises en charge par App Runner.

Versions d'exécution prises en charge : version originale d'App Runner

Nom de l’exécution	Versions mineures	Forfaits inclus
Go 1 (Go 1)	1,18,10	
	1,18,9	
	1,18,8	
	1,18,7	

> **Note**
>
> App Runner fournit un processus de compilation révisé pour des environnements d'exécution majeurs spécifiques qui ont été publiés plus récemment. Pour cette raison, vous verrez des références à la version révisée d'App Runner et à la version originale d'App Runner dans certaines sections de ce document. Pour plus d'informations, consultez Versions d'exécution gérées et build d'App Runner.

Développement du code d'application pour App Runner

Ce chapitre décrit les informations d'exécution et les directives de développement que vous devez prendre en compte lors du développement ou de la migration du code d'application à des fins de AWS App Runner déploiement.

Informations d'exécution

Que vous fournissiez une image de conteneur ou qu'App Runner en crée une pour vous, App Runner exécute le code de votre application dans une instance de conteneur. Voici quelques aspects clés de l'environnement d'exécution des instances de conteneurs.

- Support du framework : App Runner prend en charge toute image implémentant une application Web. Il est indépendant du langage de programmation que vous choisissez et du serveur ou du framework d'applications Web que vous utilisez, le cas échéant. Pour votre commodité, nous proposons des environnements d'exécution gérés spécifiques aux différentes plateformes de programmation, afin de rationaliser le processus de création d'applications et de création d'images abstraites.
- Demandes Web : App Runner prend en charge les protocoles HTTP 1.0 et HTTP 1.1 pour les instances de conteneur. Pour plus d'informations sur la configuration de votre service, consultezthe section called “Configuration”. Il n'est pas nécessaire d'implémenter la gestion du trafic sécurisé HTTPS. App Runner redirige toutes les requêtes HTTP entrantes vers les points de terminaison HTTPS correspondants. Il n'est pas nécessaire de configurer de paramètres pour permettre la redirection des requêtes Web HTTP. App Runner met fin au TLS avant de transmettre les demandes à votre instance de conteneur d'applications.

  > **Note**
  >
  > - Le délai d'expiration des requêtes HTTP est de 120 secondes au total. Les 120 secondes incluent le temps nécessaire à l'application pour lire la demande, y compris le corps, et terminer l'écriture de la réponse HTTP.
  > - Le délai limite de lecture et de réponse des demandes dépend des applications que vous utilisez. Ces applications peuvent avoir leurs propres délais d'expiration internes, comme le serveur HTTP pour Python, Gunicorn, qui a une limite de délai d'expiration par défaut de 30 secondes. Dans de tels cas, le délai d'expiration de l'application dépasse le délai d'expiration de 120 secondes d'App Runner.

- Vous n'avez pas besoin de configurer les suites de chiffrement TLS ni aucun autre paramètre, car App Runner est un service entièrement géré qui gère la terminaison du protocole TLS pour vous.

- Applications apatrides — Actuellement, App Runner ne prend pas en charge les applications statiques. Par conséquent, App Runner ne garantit pas la persistance de l'état au-delà de la durée de traitement d'une seule requête Web entrante.
- Stockage : App Runner augmente ou diminue automatiquement les instances de votre application App Runner en fonction du volume de trafic entrant. Vous pouvez configurer les options de dimensionnement automatique pour votre application App Runner. Étant donné que le nombre d'instances actuellement actives traitant les requêtes Web est basé sur le volume de trafic entrant, App Runner ne peut garantir que les fichiers puissent persister au-delà du traitement d'une seule demande. Par conséquent, App Runner implémente le système de fichiers dans votre instance de conteneur sous forme de stockage éphémère, ce qui implique que les fichiers sont transitoires. Par exemple, les fichiers ne sont pas conservés lorsque vous interrompez et reprenez votre service App Runner.

  App Runner vous fournit 3 Go de stockage éphémère et utilise une partie des 3 Go de stockage éphémère pour son image de conteneur extraite, compressée et non compressée sur l'instance. Le stockage éphémère restant peut être utilisé par votre service App Runner. Toutefois, il ne s'agit pas d'un stockage permanent en raison de son caractère apatride.

  > **Note**
  >
  > Il peut arriver que les fichiers de stockage persistent malgré les demandes. Par exemple, si la demande suivante arrive sur la même instance, les fichiers de stockage seront conservés. La persistance des fichiers de stockage entre les demandes peut être utile dans certaines situations. Par exemple, lors du traitement d'une demande, vous pouvez mettre en cache les fichiers que votre application télécharge si de futures demandes peuvent en avoir besoin. Cela peut accélérer le traitement des demandes futures, mais ne garantit pas les gains de vitesse. Votre code ne doit pas partir du principe qu'un fichier téléchargé lors d'une demande précédente existe toujours.
  > Pour garantir la mise en cache à l'aide d'un magasin de données en mémoire à haut débit et à faible latence, utilisez un service tel qu'Amazon. ElastiCache

- Variables d'environnement : par défaut, App Runner rend la variable d'`PORT`environnement disponible dans votre instance de conteneur. Vous pouvez configurer la valeur de la variable

avec les informations de port et ajouter des variables et des valeurs d'environnement personnalisées. Vous pouvez également référencer des données sensibles stockées dans AWS Secrets Managerou AWS Systems Manager Parameter Store en tant que variables d'environnement. Pour plus d'informations sur la création de variables d'environnement, consultezVariables d'environnement de référence.

- Rôle d'instance : si le code de votre application appelle un service, en utilisant le service APIs ou l'un d'entre eux AWS SDKs, créez un rôle d'instance à l'aide de Gestion des identités et des accès AWS (IAM). AWS Ensuite, attachez-le à votre service App Runner lorsque vous le créez. Incluez toutes les autorisations d'action de AWS service requises par votre code dans votre rôle d'instance. Pour de plus amples informations, veuillez consulter the section called “Rôle d’instance”.

Directives de développement du code

Tenez compte de ces directives lorsque vous développez du code pour une application Web App Runner.

- Corriger les images du conteneur : lorsque vous fournissez des images du conteneur, vous êtes responsable de les mettre à jour et de corriger régulièrement ces images. Pendant qu'App Runner gère l'infrastructure, vous devez vous assurer de la sécurité et de l' up-to-dateétat des images de conteneur fournies. Pour plus d'informations, consultez la documentation AWS App Runner
- Concevez un code apatride : concevez l'application Web que vous déployez sur votre service App Runner pour qu'elle soit apatride. Votre code doit partir du principe qu'aucun état ne persiste au-delà de la durée de traitement d'une seule requête Web entrante.
- Supprimer les fichiers temporaires : lorsque vous créez des fichiers, ils sont stockés sur un système de fichiers et occupent une partie de l'espace de stockage alloué à votre service. Pour éviter les out-of-storage erreurs, ne conservez pas les fichiers temporaires pendant de longues périodes. Équilibrez la taille du stockage avec la vitesse de traitement des demandes lorsque vous prenez des décisions relatives à la mise en cache des fichiers.
- Démarrage de l'instance : App Runner permet de démarrer l'instance en cinq minutes. Votre instance doit écouter les demandes sur ses ports d'écoute configurés et être saine dans les cinq minutes suivant son démarrage. Au démarrage, les instances App Runner se voient attribuer un processeur virtuel (vCPU) en fonction de la configuration de votre vCPU. Pour plus d'informations sur la configuration des vCPU disponibles, consultez. the section called “Configurations compatibles avec App Runner”

Une fois que l'instance a démarré avec succès, elle passe à l'état inactif et attend les demandes. Vous payez en fonction de la durée de démarrage de l'instance, avec un minimum d'une minute par démarrage d'instance. Pour plus d’informations sur la tarification, consultez Tarification AWS App Runner.

Utilisation de la console App Runner

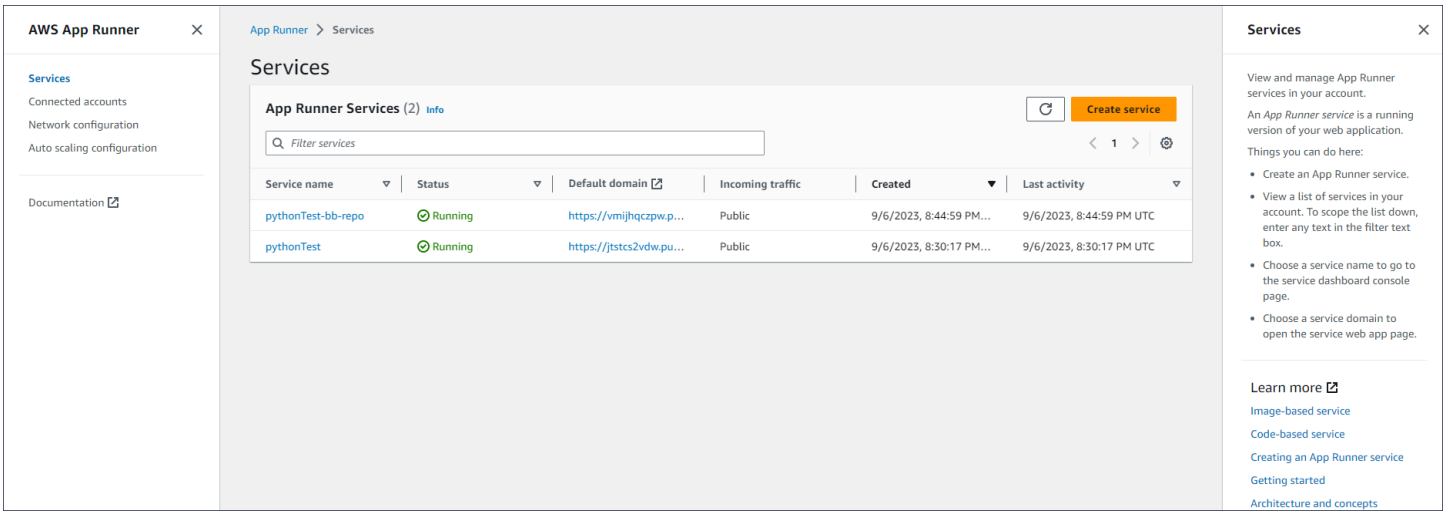
Utilisez la AWS App Runner console pour créer, gérer et surveiller vos services App Runner et les ressources associées, telles que les comptes connectés. Vous pouvez consulter les services existants, en créer de nouveaux et configurer un service. Vous pouvez consulter l'état d'un service App Runner ainsi que consulter les journaux, surveiller l'activité et suivre les statistiques. Vous pouvez également accéder au site Web de votre service ou à votre référentiel de sources.

Les sections suivantes décrivent la disposition et les fonctionnalités de la console, et vous renvoient aux informations associées.

Disposition générale de la console

La console App Runner comporte trois zones. De gauche à droite :

- Volet de navigation : volet latéral qui peut être réduit ou agrandi. Utilisez-le pour choisir la page de console de niveau supérieur que vous souhaitez utiliser.
- Volet de contenu : partie principale de la page de console. Utilisez-le pour consulter les informations et effectuer vos tâches.
- Volet d'aide : volet latéral pour plus d'informations. Développez-le pour obtenir de l'aide concernant la page sur laquelle vous vous trouvez. Vous pouvez également choisir n'importe quel lien d'information sur une page de console pour obtenir de l'aide contextuelle.



La page Services

La page Services répertorie les services App Runner de votre compte. Vous pouvez étendre la liste vers le bas à l'aide de la zone de texte du filtre.

Pour accéder à la page Services

1. Ouvrez la console App Runner, puis dans la liste des régions, sélectionnez votre Région AWS.
2. Dans le panneau de navigation, choisissez Services.

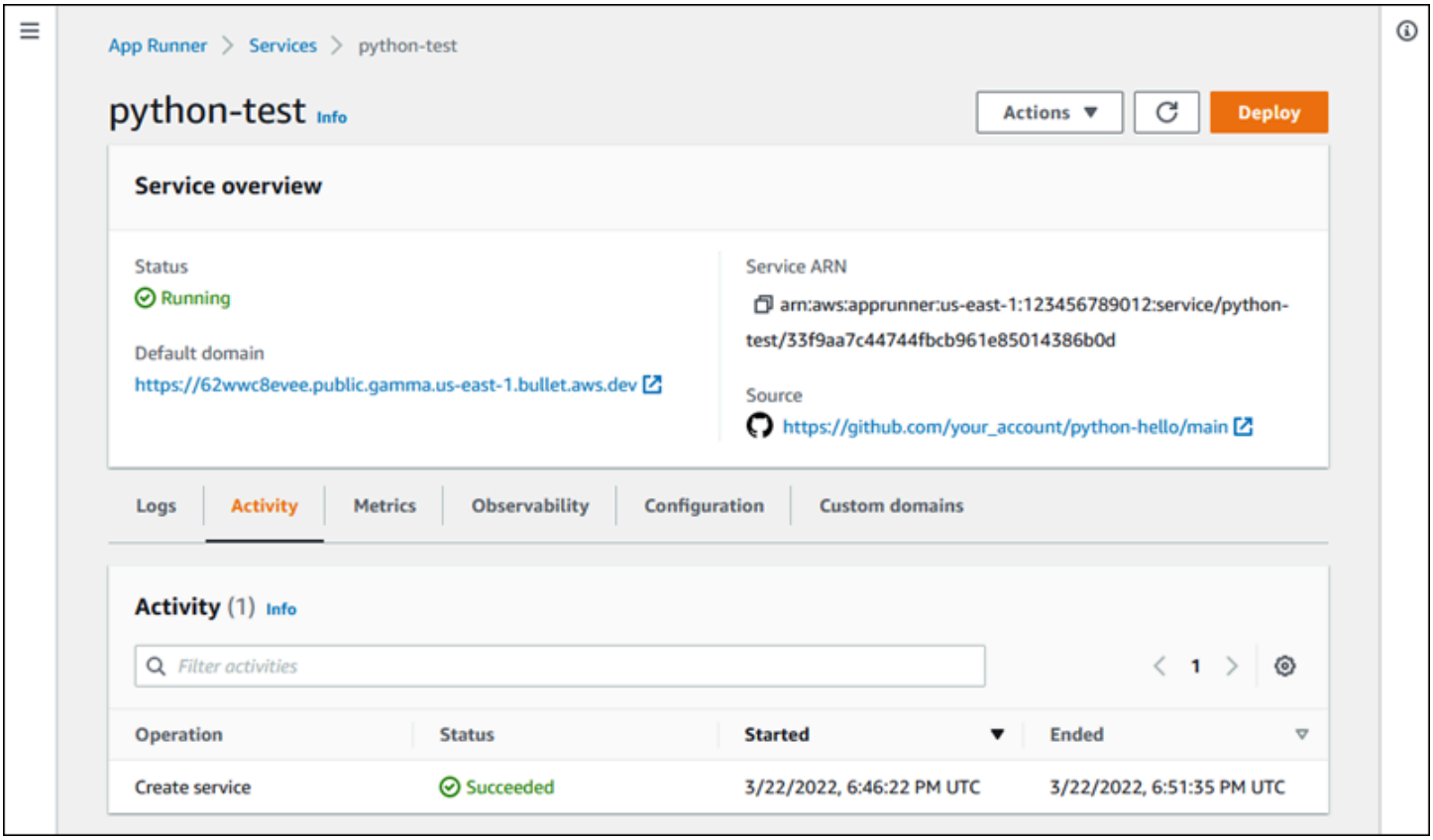
Ce que vous pouvez faire ici :

- Créez un service App Runner. Pour de plus amples informations, veuillez consulter the section called “Création”.
- Choisissez un nom de service pour accéder à la page de console du tableau de bord du service.
- Choisissez un domaine de service pour ouvrir la page de l'application Web du service.

La page du tableau de bord du service

Vous pouvez consulter les informations relatives à un service App Runner et le gérer depuis la page du tableau de bord du service. En haut de la page, vous pouvez voir le nom du service.

Pour accéder au tableau de bord des services, accédez à la page Services (voir section précédente), puis choisissez votre service App Runner.



La section Présentation du service fournit des informations de base sur le service App Runner et votre application. Ce que vous pouvez faire ici :

- Affichez les détails du service tels que l'état, l'état de santé et l'ARN.
- Accédez au domaine par défaut : le domaine fourni par App Runner pour l'application Web exécutée dans votre service. Il s'agit d'un sous-domaine du `awsapprunner.com` domaine appartenant à App Runner.
- Accédez au référentiel source déployé sur le service.
- Démarrez le déploiement d'un référentiel source sur votre service.
- Suspendez, reprenez et supprimez votre service.

Les onglets situés sous l'aperçu des services concernent la gestion et l'observabilité des services.

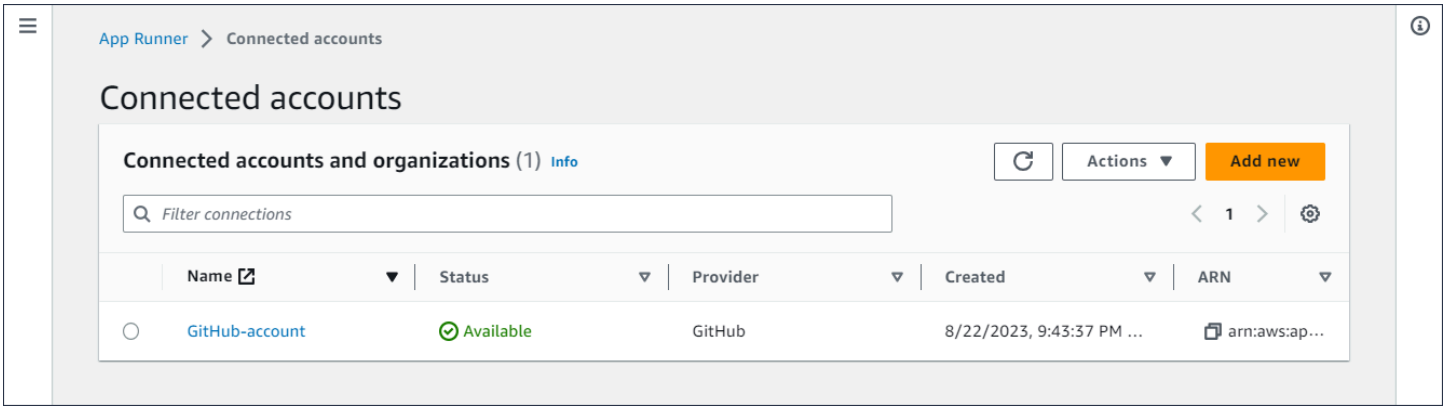
La page des comptes connectés

La page Comptes connectés répertorie les connexions App Runner aux fournisseurs de référentiels de code source de votre compte. Vous pouvez étendre la liste vers le bas à l'aide de la zone de

texte du filtre. Pour plus d'informations sur les comptes connectés, consultezthe section called "Connexions".

Pour accéder à la page Comptes connectés

1. Ouvrez la console App Runner, puis dans la liste des régions, sélectionnez votre Région AWS.
2. Dans le volet de navigation, sélectionnez Comptes connectés.



Ce que vous pouvez faire ici :

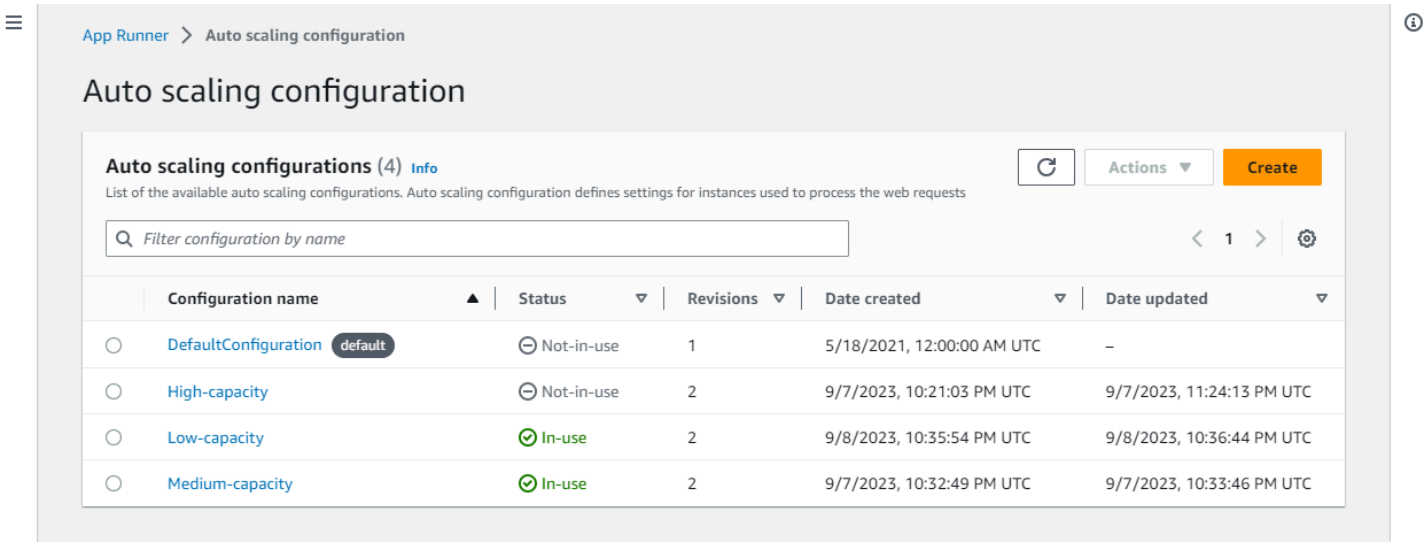
- Consultez la liste des connexions aux fournisseurs de référentiels dans votre compte. Pour étendre la liste vers le bas, entrez n'importe quel texte dans la zone de texte du filtre.
- Choisissez un nom de connexion pour accéder au compte fournisseur ou à l'organisation correspondant.
- Sélectionnez une connexion pour terminer la prise de contact pour une connexion que vous venez d'établir (dans le cadre de la création d'un service) ou pour supprimer la connexion.

La page des configurations de mise à l'échelle automatique

La page Configurations de dimensionnement automatique répertorie les configurations de dimensionnement automatique que vous avez configurées dans votre compte. Vous pouvez configurer quelques paramètres pour ajuster le comportement de mise à l'échelle automatique et les enregistrer dans différentes configurations que vous pourrez ensuite attribuer à un ou plusieurs services App Runner. Vous pouvez étendre la liste vers le bas à l'aide de la zone de texte du filtre. Pour plus d'informations sur les configurations de mise à l'échelle automatique, consultezGérer le dimensionnement automatique d'un service.

Pour accéder à la page de configuration d'Auto Scaling

1. Ouvrez la console App Runner, puis dans la liste des régions, sélectionnez votre Région AWS.
2. Dans le volet de navigation, choisissez Auto scaling configuration.



Ce que vous pouvez faire ici :

- Consultez la liste des configurations de mise à l'échelle automatique existantes dans votre compte.
- Créez une nouvelle configuration de mise à l'échelle automatique ou une révision d'une configuration existante.
- Définissez une configuration de dimensionnement automatique par défaut pour les nouveaux services que vous créez.
- Supprimez une configuration.
- Sélectionnez le nom d'une configuration pour accéder au panneau des révisions Auto Scaling afin de gérer les révisions.

Gestion de votre service App Runner

Ce chapitre décrit comment gérer votre AWS App Runner service. Dans ce chapitre, vous apprendrez à gérer le cycle de vie de votre service : créer, configurer et supprimer un service, déployer de nouvelles versions d'applications sur votre service et contrôler la disponibilité de votre service Web en interrompant puis en reprenant le service. Vous apprendrez également à gérer d'autres aspects de votre service, tels que les connexions et le dimensionnement automatique.

Rubriques

- Création d'un service App Runner
- Reconstruction d'un service App Runner défaillant
- Déploiement d'une nouvelle version de l'application sur App Runner
- Configuration d'un service App Runner
- Gestion des connexions App Runner
- Gestion du dimensionnement automatique d'App Runner
- Gestion des noms de domaine personnalisés pour un service App Runner
- Suspension et reprise d'un service App Runner
- Supprimer un service App Runner

Création d'un service App Runner

AWS App Runner automatise la transition d'une image de conteneur ou d'un référentiel de code source vers un service Web en cours d'exécution qui évolue automatiquement. Vous pointez App Runner vers votre image ou votre code source, en spécifiant uniquement un petit nombre de paramètres requis. App Runner construit votre application si nécessaire, provisionne des ressources de calcul et déploie votre application pour qu'elle s'exécute sur celles-ci.

Lorsque vous créez un service, App Runner crée une ressource de service. Dans certains cas, vous devrez peut-être fournir une ressource de connexion. Si vous utilisez la console App Runner, celle-ci crée implicitement la ressource de connexion. Pour plus d'informations sur les types de ressources App Runner, consultezthe section called “Ressources pour App Runner”. Pour chacun de ces types de ressources, des quotas sont associés à votre compte Région AWS. Pour de plus amples informations, veuillez consulter the section called “Quotas de ressources App Runner”.

Il existe des différences subtiles dans la procédure de création d'un service en fonction du type de source et du fournisseur. Cette rubrique décrit les différentes procédures de création de ces types de sources afin que vous puissiez suivre celle qui convient le mieux à votre situation. Pour démarrer une procédure de base avec un exemple de code, consultezPrise en main.

Conditions préalables

Avant de créer votre service App Runner, assurez-vous d'effectuer les actions suivantes :

- Effectuez les étapes de configuration dansConfiguration.
- Assurez-vous que le code source de votre application est prêt. Vous pouvez utiliser un référentiel de code dans GitHubBitbucket ou une image de conteneur dans Amazon Elastic Container Registry (Amazon ECR) pour créer un service App Runner.

Créer un service

Cette section décrit le processus de création des deux types de service App Runner : basé sur le code source et basé sur une image de conteneur.

> **Note**
>
> Si vous créez un connecteur VPC pour le trafic sortant pour un service, le processus de démarrage du service qui suit connaîtra une latence unique. Vous pouvez définir cette configuration pour un nouveau service lorsque vous le créez, ou ultérieurement, avec une mise à jour du service. Pour plus d'informations, consultez le Latence unique chapitre de ce guide consacré à la mise en réseau avec App Runner.

Création d'un service à partir d'un référentiel de code

Les sections suivantes montrent comment créer un service App Runner lorsque votre source est un dépôt de code dans GitHubou Bitbucket. Lorsque vous utilisez un référentiel de code, App Runner doit se connecter à l'organisation ou au compte du fournisseur. Par conséquent, vous devez aider à établir cette connexion. Pour plus d'informations sur les connexions App Runner, consultezthe section called “Connexions”.

Lorsque vous créez le service, App Runner crée une image Docker contenant le code de votre application et ses dépendances. Il lance ensuite un service qui exécute une instance de conteneur de cette image.

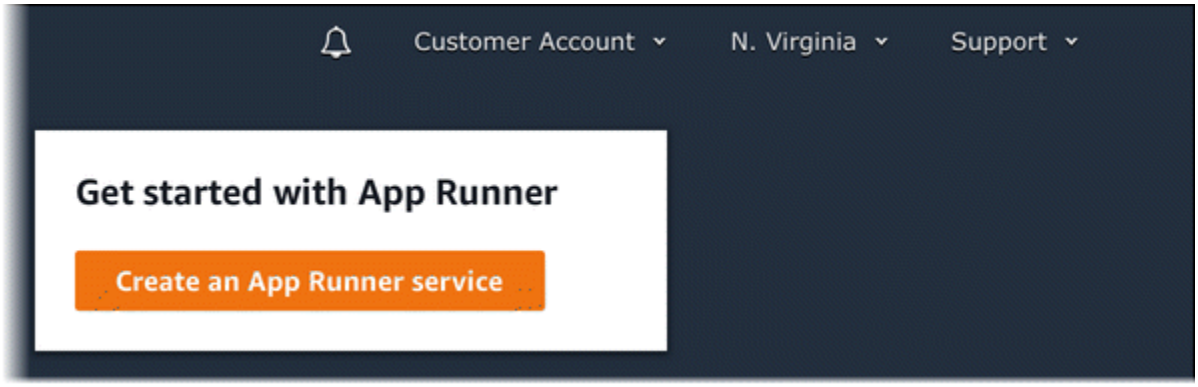
Création d'un service à partir du code à l'aide de la console App Runner

Pour créer un service App Runner à l'aide de la console

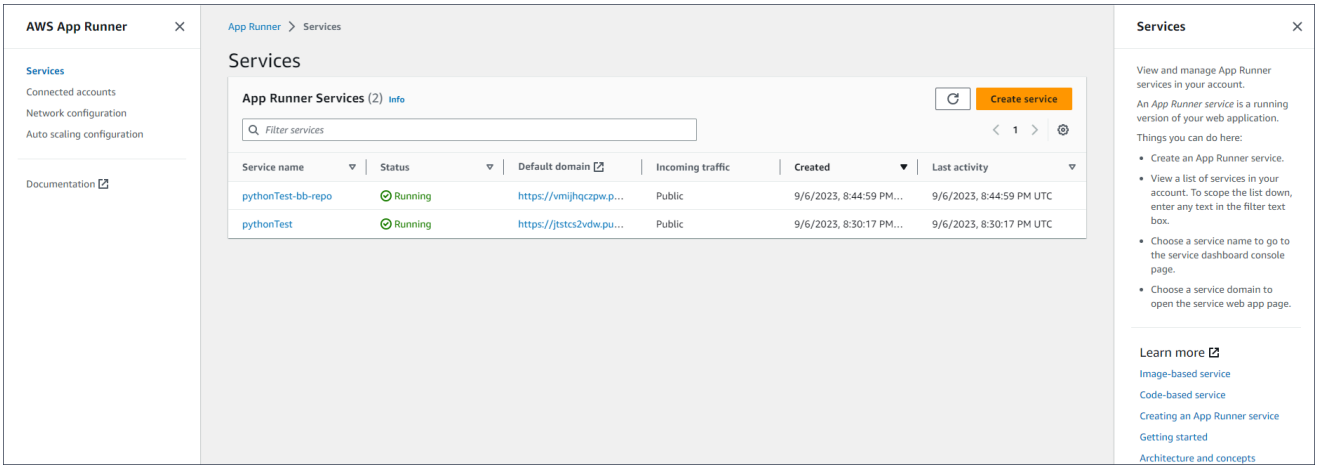
1. Configurez votre code source.

   a. Ouvrez la console App Runner, puis dans la liste des régions, sélectionnez votre Région AWS.

   b. Si aucun service App Runner Compte AWS n'est encore disponible, la page d'accueil de la console s'affiche. Choisissez Créer un service App Runner.



   S'il Compte AWS possède des services existants, la page Services contenant la liste de vos services s'affiche. Choisissez Créer un service.



   c. Sur la page Source et déploiement, dans la section Source, pour Type de référentiel, choisissez Référentiel de code source.

   d. Sélectionnez un type de fournisseur. Choisissez l'un GitHubou l'autre ou Bitbucket.

   e. Sélectionnez ensuite un compte ou une organisation pour le fournisseur que vous avez utilisé auparavant, ou choisissez Ajouter un nouveau. Ensuite, entrez les informations

d'identification de votre référentiel de code et choisissez un compte ou une organisation auxquels vous connecter.

f. Pour Repository, sélectionnez le référentiel qui contient le code de votre application.

g. Pour Branch, sélectionnez la branche que vous souhaitez déployer.

h. Dans le champ Répertoire source, entrez le répertoire du référentiel source qui stocke le code de votre application et les fichiers de configuration.

> **Note**
>
> Les commandes build et start s'exécutent à partir du répertoire source que vous spécifiez. App Runner gère le chemin comme absolu depuis le root. Si vous ne spécifiez aucune valeur ici, le répertoire prend par défaut la racine du dépôt.

2. Configurez vos déploiements.

a. Dans la section Paramètres de déploiement, choisissez Manuel ou Automatique.

Pour plus d'informations sur les méthodes de déploiement, consultezthe section called “Méthodes de déploiement”.

b. Choisissez Suivant.

Source and deployment Info

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

Source and deployment

Source

Repository type

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

Provider

Choose the provider where you host your code repository.

GitHub

Github Connection Info

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

myGitHub | Add new

Repository

python-hello

Branch

main

Source directory

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

/

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

Deployment settings

Deployment trigger

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

Cancel Next

3. Configurez le build de l'application.

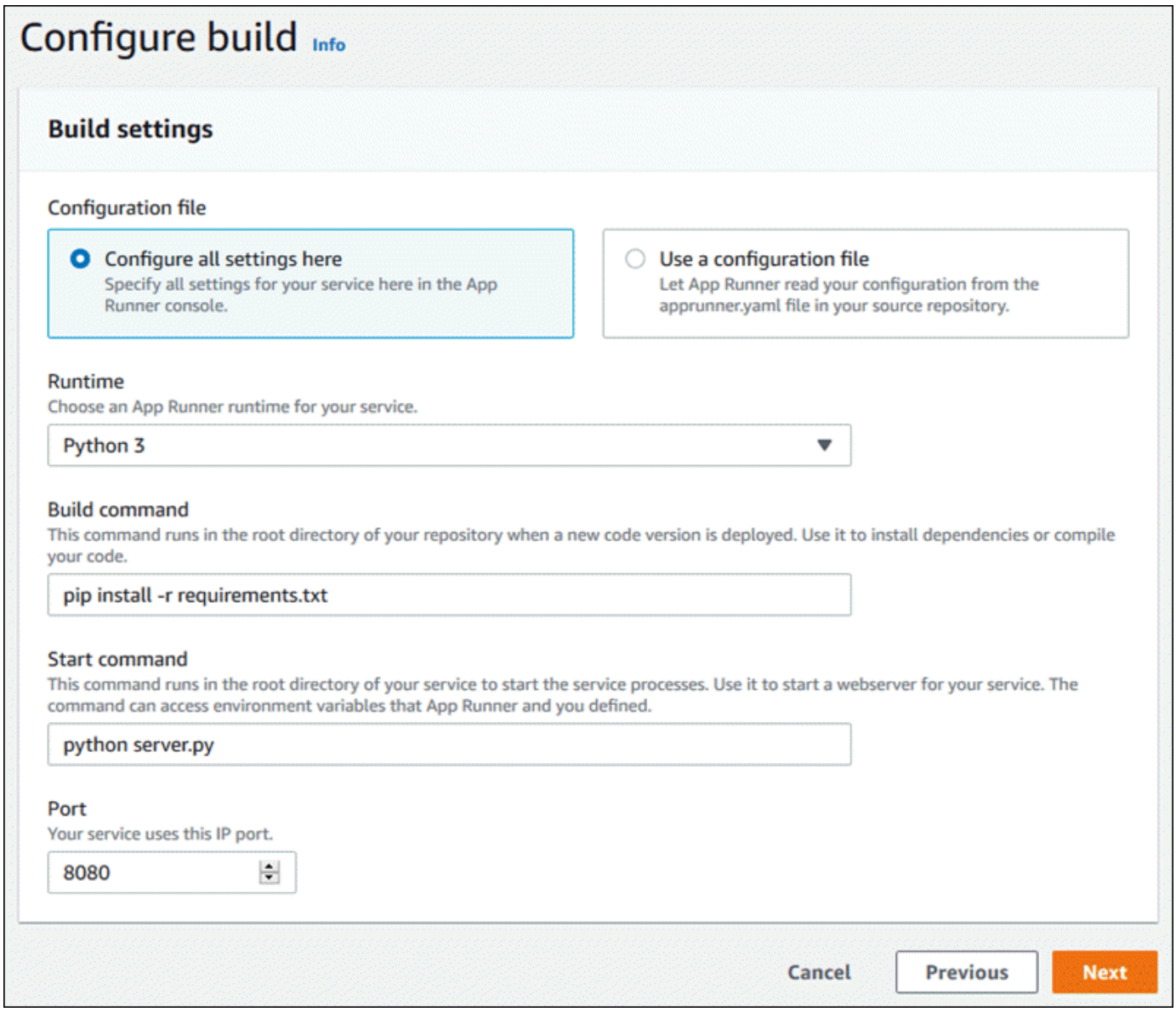
   a. Sur la page Configurer le build, pour Fichier de configuration, choisissez Configurer tous les paramètres ici si votre référentiel ne contient pas de fichier de configuration App Runner, ou Utiliser un fichier de configuration s'il en contient un.

      > **Note**
      >
      > Un fichier de configuration App Runner permet de conserver la configuration de votre build dans le cadre de la source de votre application. Lorsque vous en fournissez une, App Runner lit certaines valeurs du fichier et ne vous permet pas de les définir dans la console.

   b. Fournissez les paramètres de compilation suivants :

      - Runtime — Choisissez un environnement d'exécution géré spécifique pour votre application.
      - Commande de génération : entrez une commande qui crée votre application à partir de son code source. Il peut s'agir d'un outil spécifique au langage ou d'un script fourni avec votre code.
      - Commande de démarrage : entrez la commande qui démarre votre service Web.
      - Port — Entrez le port IP écouté par votre service Web.

   c. Choisissez Suivant.



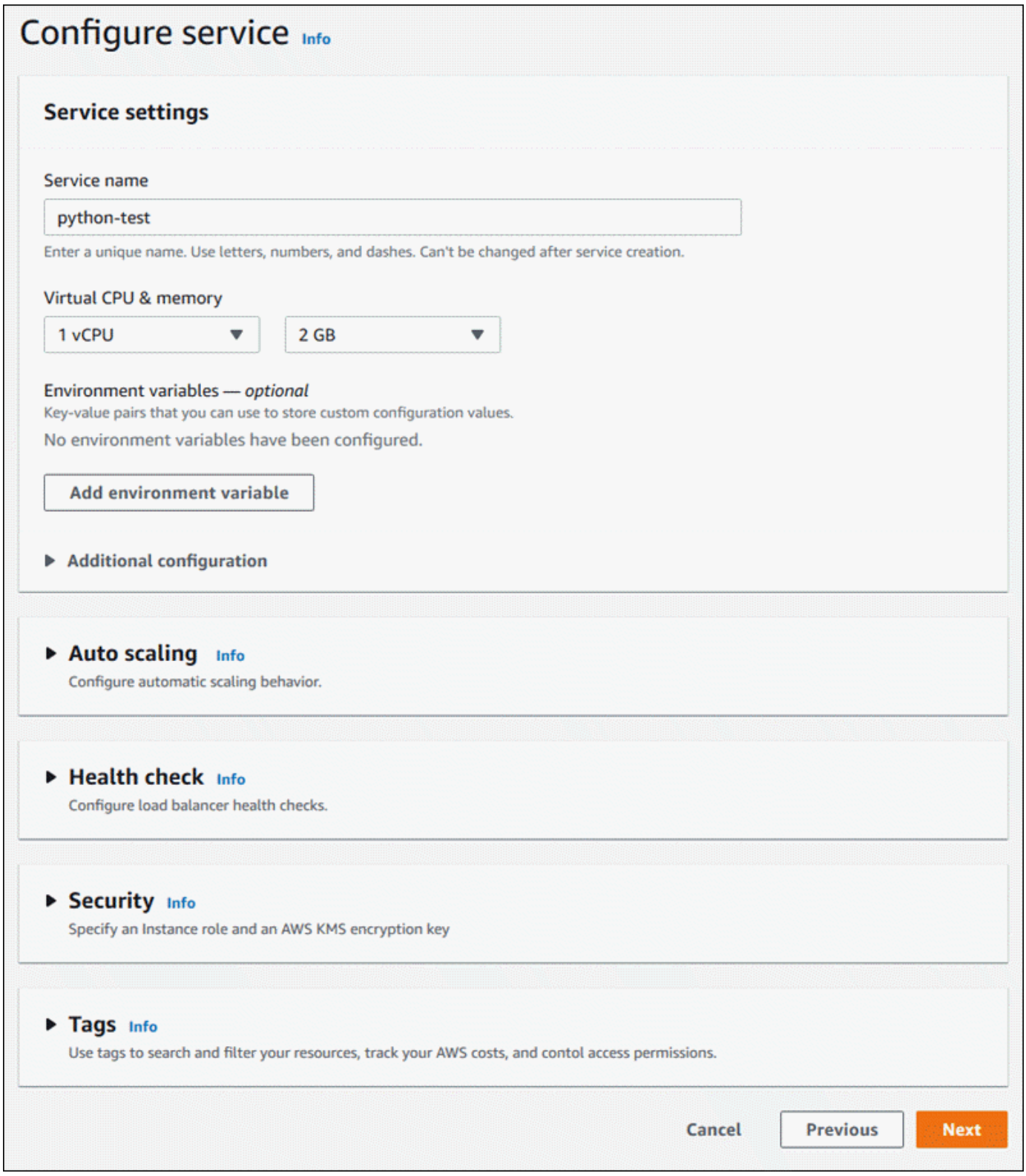
4. Configurez votre service.

   a. Sur la page Configurer le service, dans la section Paramètres du service, entrez un nom de service.

> **Note**
>
> Tous les autres paramètres de service sont facultatifs ou comportent des paramètres par défaut fournis par la console.

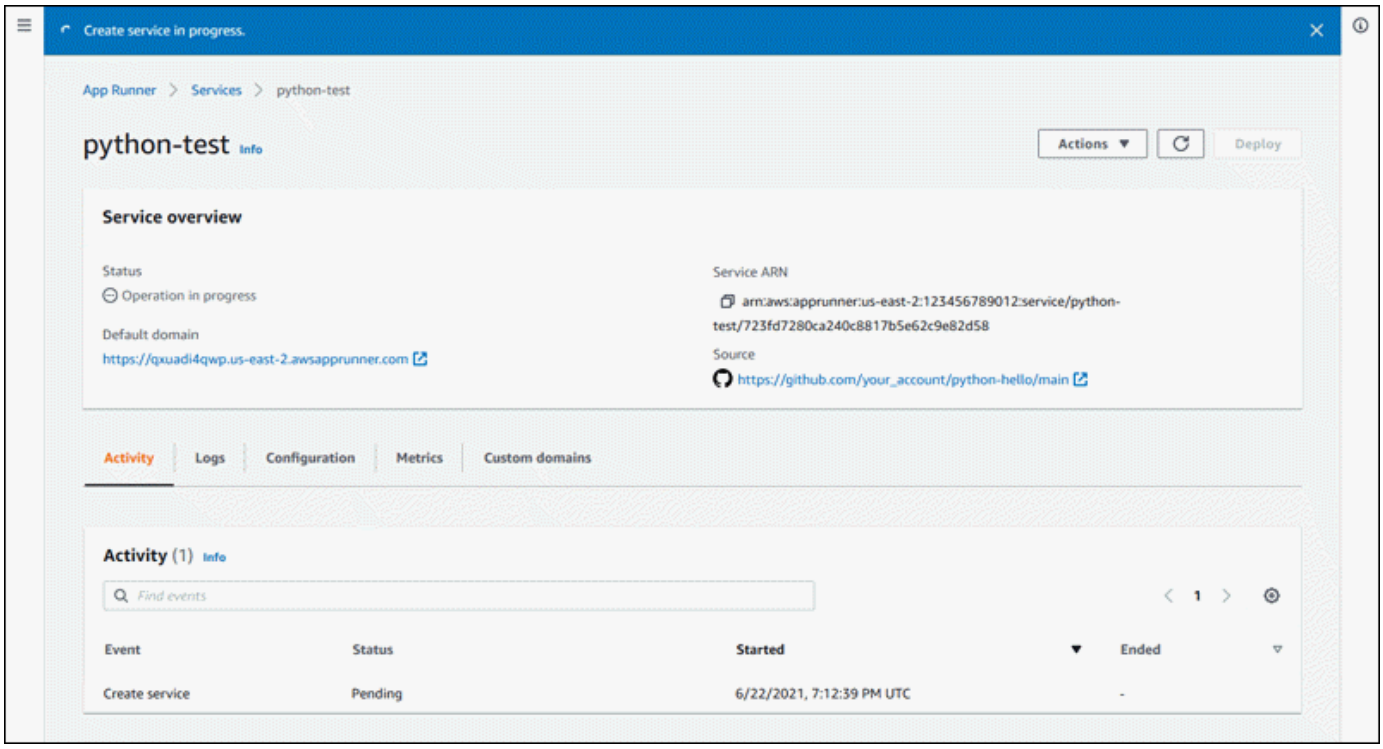
b. Modifiez ou ajoutez éventuellement d'autres paramètres pour répondre aux exigences de votre application.

c. Choisissez Suivant.



5. Sur la page Réviser et créer, vérifiez tous les détails que vous avez saisis, puis choisissez Créer et déployer.

Résultat : Si le service est créé avec succès, la console affiche le tableau de bord du service avec une vue d'ensemble du nouveau service.



6. Vérifiez que votre service est en cours d'exécution.

   a. Sur la page du tableau de bord du service, attendez que le statut du service soit en cours d'exécution.

   b. Choisissez la valeur de domaine par défaut. Il s'agit de l'URL du site Web de votre service.

   c. Utilisez votre site Web et vérifiez qu'il fonctionne correctement.

Création d'un service à partir du code à l'aide de l'API App Runner ou AWS CLI

Pour créer un service à l'aide de l'API App Runner ou AWS CLI appelez l'action `CreateService` API. Pour plus d'informations et pour voir un exemple, consultez CreateService. Si c'est la première fois que vous créez un service à l'aide d'une organisation ou d'un compte spécifique pour un référentiel de code source (GitHub ou Bitbucket), commencez par appeler CreateConnection. Cela établit une connexion entre App Runner et l'organisation ou le compte du fournisseur du référentiel. Pour plus d'informations sur les connexions App Runner, consultezthe section called “Connexions”.

Si l'appel renvoie une réponse positive avec un objet Service affiché`"Status": "CREATING"`, votre service commence à être créé.

Pour un exemple d'appel, voir Créer un service de référentiel de code source dans le Guide de référence des AWS App Runner API

Création d'un service à partir d'une image Amazon ECR

Les sections suivantes montrent comment créer un service App Runner lorsque votre source est une image de conteneur stockée dans Amazon ECR. Amazon ECR est un Service AWS. Par conséquent, pour créer un service basé sur une image Amazon ECR, vous fournissez à App Runner un rôle d'accès contenant les autorisations d'action Amazon ECR nécessaires.

> **Note**
>
> Les images stockées dans Amazon ECR Public sont accessibles au public. Ainsi, si votre image est stockée dans Amazon ECR Public, aucun rôle d'accès n'est requis.

Lors de la création de votre service, App Runner lance un service qui exécute une instance de conteneur de l'image que vous fournissez. Il n'y a pas de phase de construction dans ce cas.

Pour de plus amples informations, veuillez consulter Service basé sur l'image.

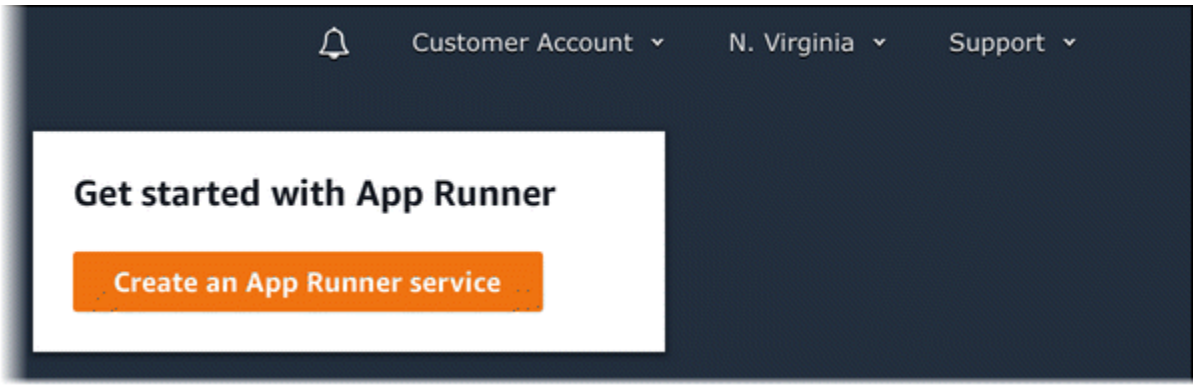
Création d'un service à partir d'une image à l'aide de la console App Runner

Pour créer un service App Runner à l'aide de la console

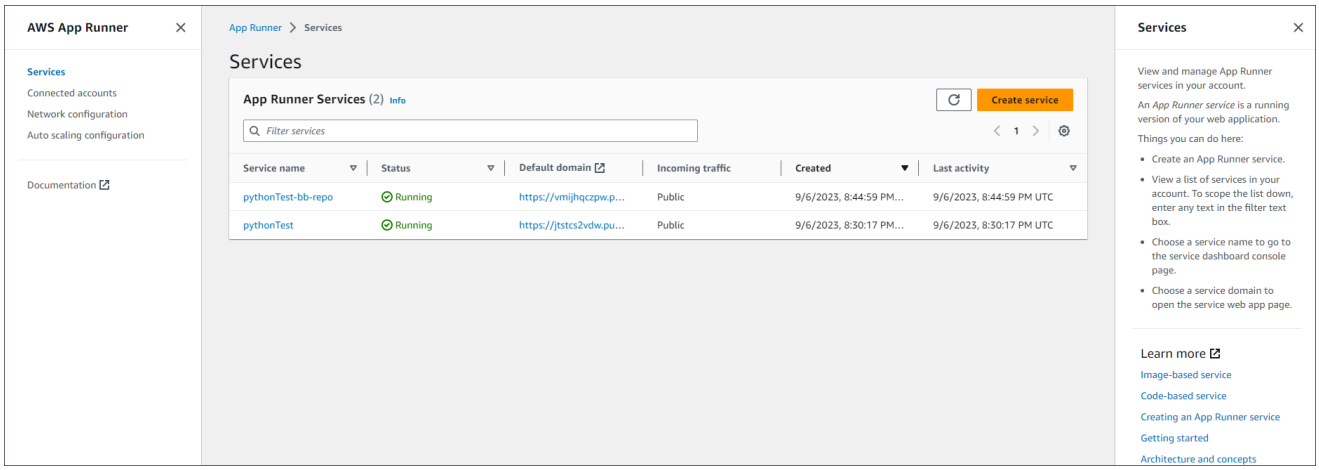
1. Configurez votre code source.

   a. Ouvrez la console App Runner, puis dans la liste des régions, sélectionnez votre Région AWS.

   b. Si aucun service App Runner Compte AWS n'est encore disponible, la page d'accueil de la console s'affiche. Choisissez Créer un service App Runner.



S'il Compte AWS possède des services existants, la page Services contenant la liste de vos services s'affiche. Choisissez Créer un service.



c. Sur la page Source et déploiement, dans la section Source, pour Type de référentiel, choisissez Registre de conteneurs.

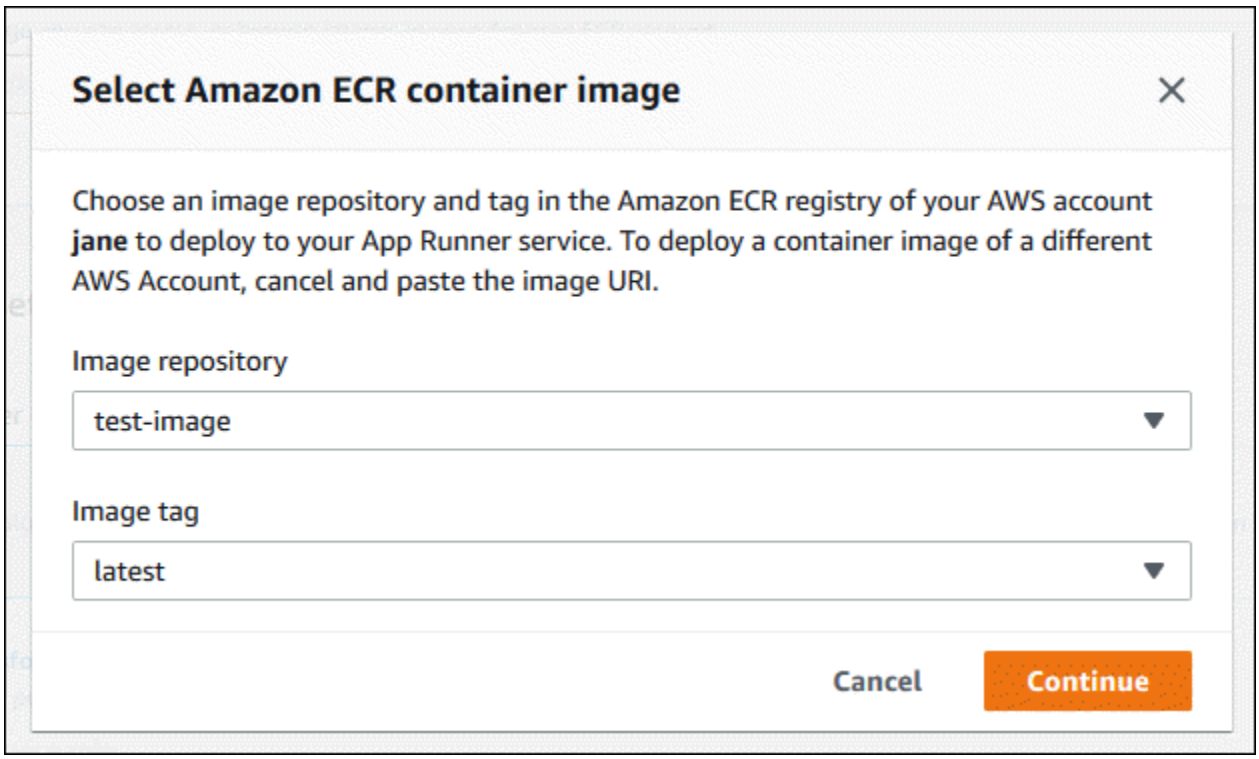
d. Dans le champ Fournisseur, choisissez le fournisseur dans lequel votre image est stockée :

- Amazon ECR : image privée stockée dans Amazon ECR.
- Amazon ECR Public : image lisible par le public qui est stockée dans Amazon ECR Public.

e. Pour l'URI de l'image du conteneur, choisissez Browse.

f. Dans la boîte de dialogue Sélectionner une image de conteneur Amazon ECR, pour Référentiel d'images, sélectionnez le référentiel qui contient votre image.

g. Pour Balise d'image, sélectionnez la balise d'image spécifique que vous souhaitez déployer (par exemple, la plus récente), puis choisissez Continuer.



2. Configurez vos déploiements.

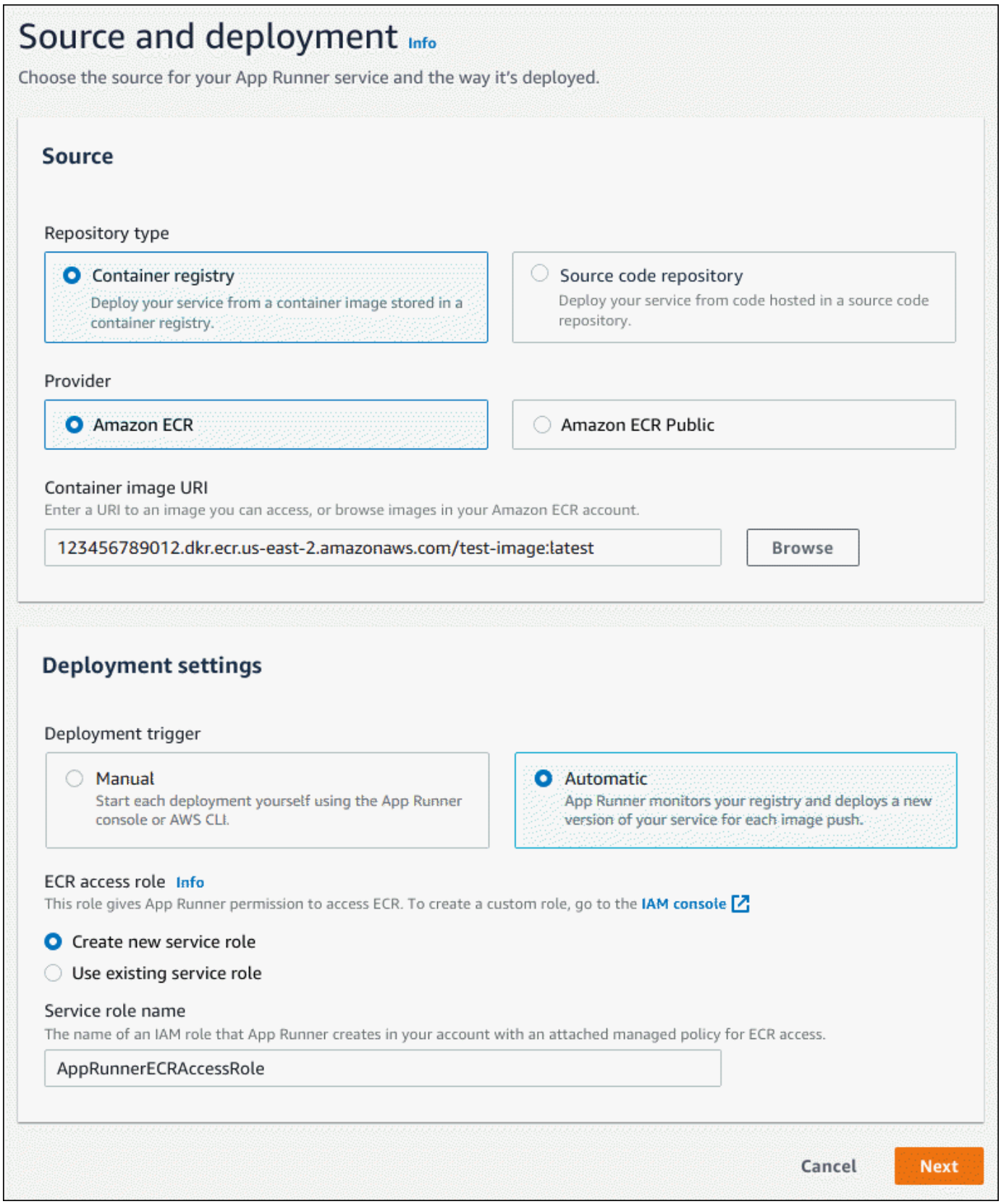
   a. Dans la section Paramètres de déploiement, choisissez Manuel ou Automatique.

   > **Note**
   >
   > App Runner ne prend pas en charge le déploiement automatique pour les images Amazon ECR Public, ni pour les images d'un référentiel Amazon ECR appartenant à un AWS compte différent de celui dans lequel se trouve votre service.

   Pour plus d'informations sur les méthodes de déploiement, consultezthe section called “Méthodes de déploiement”.

   b. [Fournisseur Amazon ECR] Pour le rôle d'accès ECR, choisissez un rôle de service existant dans votre compte ou créez un nouveau rôle. Si vous utilisez le déploiement manuel, vous pouvez également choisir d'utiliser le rôle d'utilisateur IAM au moment du déploiement.

   c. Choisissez Suivant.


Source and deployment Info
Choose the source for your App Runner service and the way it's deployed.
Source
Repository type
Container registry
Deploy your service from a container image stored in a container registry.
Source code repository
Deploy your service from code hosted in a source code repository.
Provider
Amazon ECR
Amazon ECR Public
Container image URI
Enter a URI to an image you can access, or browse images in your Amazon ECR account.
123456789012.dkr.ecr.us-east-2.amazonaws.com/test-image:latest
Browse
Deployment settings
Deployment trigger
Manual
Start each deployment yourself using the App Runner console or AWS CLI.
Automatic
App Runner monitors your registry and deploys a new version of your service for each image push.
ECR access role Info
This role gives App Runner permission to access ECR. To create a custom role, go to the IAM console
Create new service role
Use existing service role
Service role name
The name of an IAM role that App Runner creates in your account with an attached managed policy for ECR access.
AppRunnerECRAccessRole
Cancel
Next

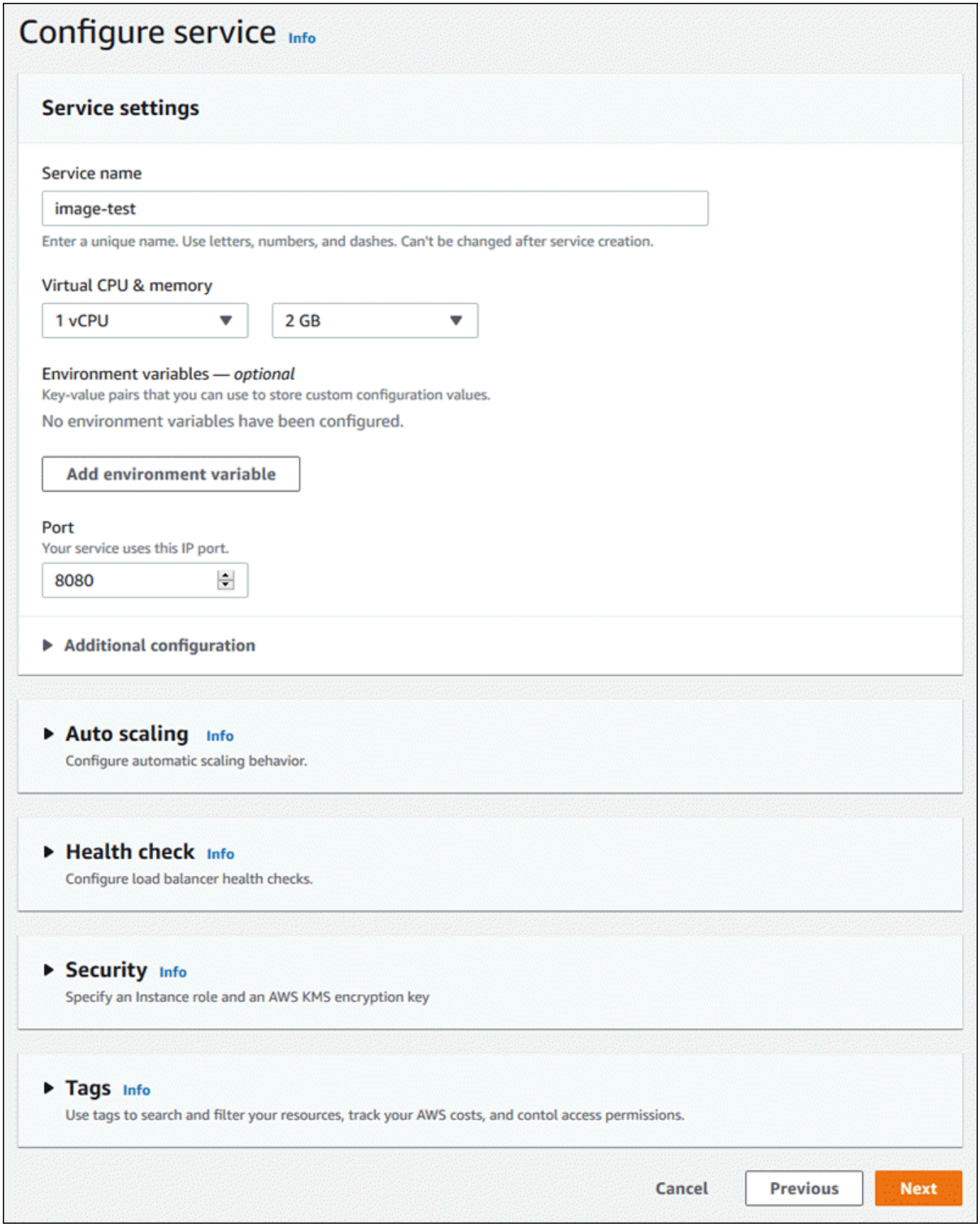
3. Configurez votre service.

   a. Sur la page Configurer le service, dans la section Paramètres du service, entrez le nom du service et le port IP écoutés par le site Web de votre service.

   > **Note**
   >
   > Tous les autres paramètres de service sont facultatifs ou comportent des paramètres par défaut fournis par la console.

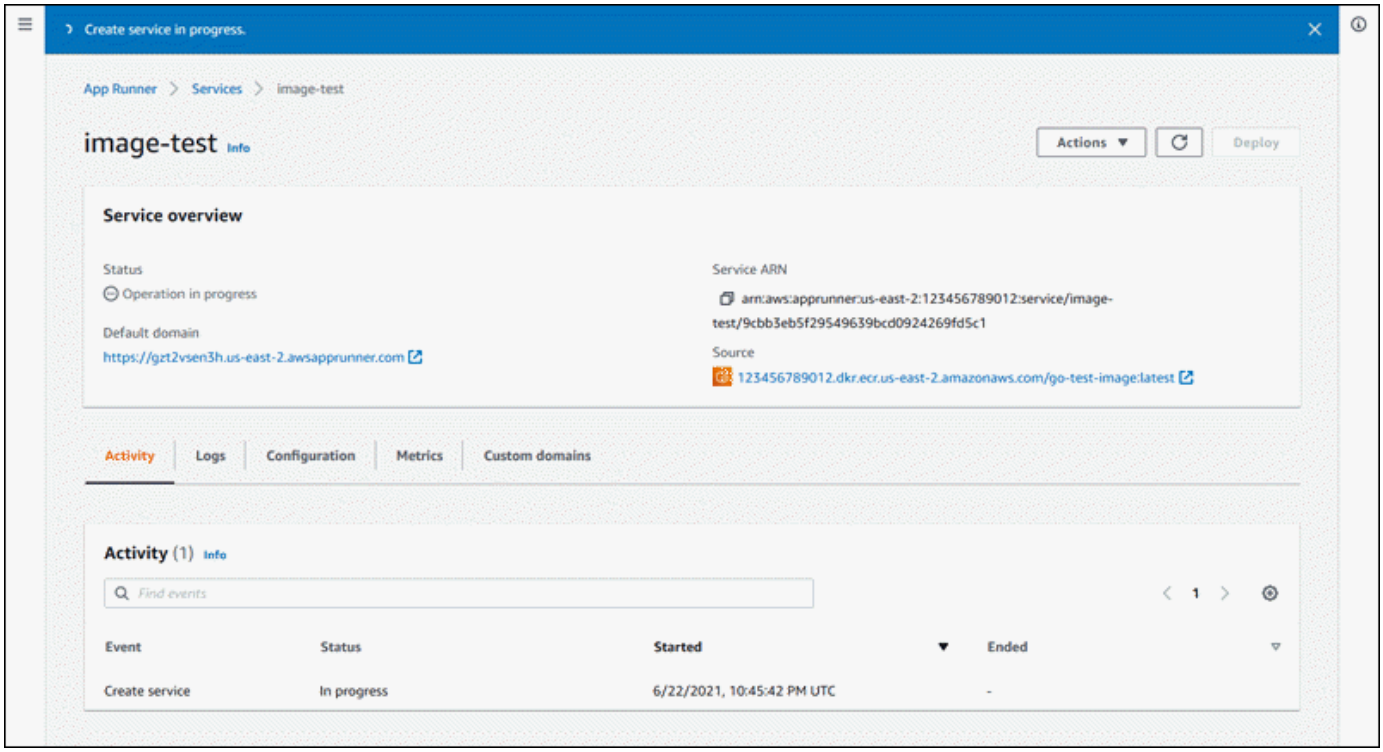
   b. (Facultatif) Modifiez ou ajoutez d'autres paramètres en fonction des besoins de votre application.

   c. Choisissez Suivant.


Configure service Info
Service settings
Service name
image-test
Enter a unique name. Use letters, numbers, and dashes. Can't be changed after service creation.
Virtual CPU & memory
1 vCPU
2 GB
Environment variables — optional
Key-value pairs that you can use to store custom configuration values.
No environment variables have been configured.
Add environment variable
Port
Your service uses this IP port.
8080
Additional configuration
Auto scaling Info
Configure automatic scaling behavior.
Health check Info
Configure load balancer health checks.
Security Info
Specify an Instance role and an AWS KMS encryption key
Tags Info
Use tags to search and filter your resources, track your AWS costs, and contol access permissions.
Cancel
Previous
Next

4. Sur la page Réviser et créer, vérifiez tous les détails que vous avez saisis, puis choisissez Créer et déployer.

   Résultat : Si le service est créé avec succès, la console affiche le tableau de bord du service, avec un aperçu du nouveau service.



5. Vérifiez que votre service fonctionne.

   a. Sur la page du tableau de bord du service, attendez que le statut du service soit en cours d'exécution.

   b. Choisissez la valeur de domaine par défaut. Il s'agit de l'URL du site Web de votre service.

   c. Utilisez votre site Web et vérifiez qu'il fonctionne correctement.

Création d'un service à partir d'une image à l'aide de l'API App Runner ou AWS CLI

Pour créer un service à l'aide de l'API App Runner ou AWS CLI appelez l'action CreateServiceAPI.

La création de votre service commence si l'appel renvoie une réponse réussie avec un objet Service affiché`"Status": "CREATING"`.

Pour un exemple d'appel, voir Créer un service de référentiel d'images source dans le Guide de référence de l'AWS App Runner API

Reconstruction d'un service App Runner défaillant

Si le message d'erreur « Impossible de créer » s'affiche lors de la création d'un service App Runner, vous pouvez effectuer l'une des opérations suivantes.

- Suivez les étapes décrites the section called “Impossible de créer le service” pour identifier la cause de l'erreur.
- Si vous trouvez une erreur dans la source ou dans la configuration, apportez les modifications nécessaires, puis reconstruisez votre service.
- Si un problème temporaire lié à App Runner a entraîné l'échec de votre service, reconstruisez le service défaillant sans apporter de modifications à la source ou à la configuration.

Vous pouvez reconstruire votre service défaillant via la console App Runner ou l'API App Runner ou AWS CLI.

Reconstruction d'un service App Runner défaillant à l'aide de la console App Runner

Rebuild with updates

La création d'un service peut échouer pour diverses raisons. Dans ce cas, il est important d'identifier et de corriger la cause première du problème avant de rétablir votre service. Pour de plus amples informations, veuillez consulter the section called “Impossible de créer le service”.

Pour reconstruire un service défaillant à l'aide de mises à jour

1. Accédez à l'onglet Configurations de votre page de service et choisissez Modifier.

   La page ouvre un panneau récapitulatif qui affiche la liste de toutes vos mises à jour.
2. Apportez les modifications requises et passez-les en revue dans le panneau récapitulatif.
3. Choisissez Enregistrer et reconstruire.

   Vous pouvez suivre les progrès dans l'onglet Logs de votre page de service.

Rebuild without updates

Si un problème temporaire entraîne l'échec de la création de votre service, vous pouvez le reconstruire sans modifier sa source ou ses paramètres de configuration.

Pour reconstruire un service défaillant sans mise à jour

- Choisissez Reconstruire dans le coin supérieur droit de votre page de service.

  Vous pouvez suivre les progrès dans l'onglet Logs de votre page de service.
- Si votre service ne parvient pas à se créer à nouveau, suivez les instructions de dépannage indiquées dansthe section called “Impossible de créer le service”. Apportez les modifications nécessaires, puis reconstruisez votre service.

Reconstruction du service App Runner défaillant à l'aide de l'API App Runner ou AWS CLI

Rebuild with updates

Pour reconstruire un service défaillant :

1. Suivez les instructions the section called “Impossible de créer le service” pour trouver la cause de l'erreur.
2. Apportez les modifications nécessaires à la branche ou à l'image du référentiel source ou à la configuration à l'origine de l'erreur.
3. Reconstruisez en appelant l'action UpdateServiceAPI avec les paramètres du nouveau référentiel de code source ou du nouveau référentiel d'images source. App Runner récupère le dernier commit depuis le dépôt du code source.

Example Reconstruction avec mises à jour

Dans l'exemple suivant, la configuration source d'un service basé sur des images est mise à jour. La valeur de `Port` est remplacée par`80`.

Mise à jour du `input.json` fichier pour le service App Runner basé sur des images

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

```
      }
    }
  }
}
```

Appel de l'action `UpdateService` API.

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

Rebuild without updates

Pour reconstruire votre service défaillant à l'aide de l'API App Runner ou AWS CLI appelez l'action UpdateServiceAPI sans modifier la source ou la configuration de votre service. Choisissez de reconstruire sans effectuer de mises à jour uniquement si la création de votre service a échoué en raison d'un problème temporaire avec App Runner.

Déploiement d'une nouvelle version de l'application sur App Runner

Lorsque vous créez un service dans AWS App Runner, vous configurez une source d'application : une image de conteneur ou un référentiel source. App Runner fournit des ressources pour exécuter votre service et y déploie votre application.

Cette rubrique décrit comment redéployer la source de votre application vers votre service App Runner lorsqu'une nouvelle version est disponible. Il peut s'agir d'une nouvelle version d'image dans le référentiel d'images ou d'un nouveau commit dans le référentiel de code. App Runner propose deux méthodes de déploiement sur un service : automatique et manuel.

Méthodes de déploiement

App Runner propose les méthodes suivantes pour vous permettre de contrôler la manière dont les déploiements d'applications sont initiés.

Déploiement automatique

Utilisez le déploiement automatique lorsque vous souhaitez un comportement d'intégration et de déploiement continus (CI/CD) pour votre service. App Runner surveille les modifications apportées à votre référentiel d'images ou de code.

Référentiel d'images : chaque fois que vous publiez une nouvelle version d'image dans votre référentiel d'images ou une nouvelle validation dans votre référentiel de code, App Runner la déploie automatiquement sur votre service sans autre action de votre part.

Référentiel de code : chaque fois que vous envoyez un nouveau commit à votre dépôt de code qui modifie le répertoire source, App Runner déploie l'intégralité de votre référentiel. Étant donné que seules les modifications apportées au répertoire source déclenchent un déploiement automatique, il est important de comprendre comment l'emplacement du répertoire source affecte l'étendue d'un déploiement automatique.

- Répertoire de premier niveau (racine du dépôt) : il s'agit de la valeur par défaut définie pour le répertoire source lorsque vous créez un service. Si votre répertoire source est défini sur cette valeur, cela signifie que l'ensemble du référentiel se trouve dans le répertoire source. Ainsi, tous les commits que vous envoyez au référentiel source déclencheront un déploiement dans ce cas.
- Tout chemin de répertoire autre que la racine du référentiel (autre que celui par défaut) : étant donné que seules les modifications transmises dans le répertoire source déclencheront un déploiement automatique, les modifications transmises à votre référentiel qui ne se trouvent pas dans le répertoire source ne déclencheront pas de déploiement automatique. Par conséquent, vous devez utiliser un déploiement manuel pour déployer les modifications que vous envoyez en dehors du répertoire source.

> **Note**
>
> App Runner ne prend pas en charge le déploiement automatique pour les images Amazon ECR Public, ni pour les images d'un référentiel Amazon ECR appartenant à un AWS compte différent de celui dans lequel se trouve votre service.

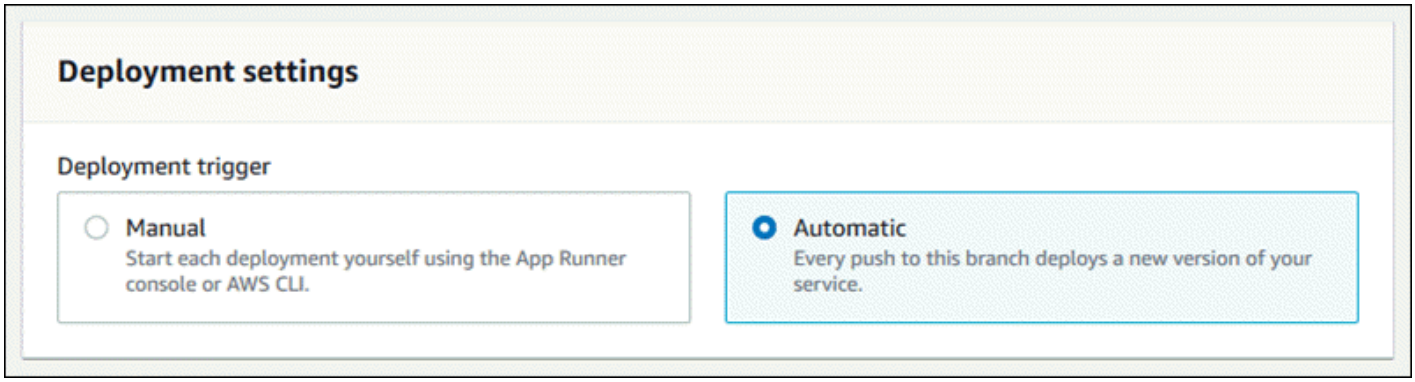
Déploiement manuel

Utilisez le déploiement manuel lorsque vous souhaitez lancer explicitement chaque déploiement sur votre service. Vous lancez un déploiement si le référentiel que vous avez configuré pour votre service contient une nouvelle version que vous souhaitez déployer. Pour de plus amples informations, veuillez consulter the section called “Déploiement manuel”.

> **Note**
>
> Lorsque vous exécutez un déploiement manuel, App Runner déploie le code source à partir du référentiel complet.

Vous pouvez configurer la méthode de déploiement de votre service de la manière suivante :

- Console : pour un nouveau service que vous créez ou pour un service existant, dans la section Paramètres de déploiement de la page Source et configuration du déploiement, choisissez Manuel ou Automatique.



- API ou AWS CLI — Dans un appel à l'UpdateServiceaction CreateServiceou, définissez le `AutoDeploymentsEnabled` membre du SourceConfigurationparamètre sur `False` pour un déploiement manuel ou `True` pour un déploiement automatique.

> **Comparaison des déploiements automatiques et manuels**
>
> Les déploiements automatiques et manuels produisent le même résultat : les deux méthodes déploient le référentiel complet.
> La différence entre les deux méthodes réside dans le mécanisme de déclenchement :
>
> - Les déploiements manuels sont déclenchés par un déploiement depuis la console, un appel ou un appel à l'API App Runner. AWS CLI La Déploiement manuel section qui suit décrit les procédures correspondantes.
> - Les déploiements automatiques sont déclenchés par une modification du contenu du répertoire source.

Déploiement manuel

Dans le cas d'un déploiement manuel, vous devez lancer explicitement chaque déploiement sur votre service. Lorsqu'une nouvelle version de l'image ou du code de votre application est prête à être déployée, vous pouvez consulter les sections suivantes pour savoir comment effectuer un déploiement à l'aide de la console et de l'API.

> **Note**
>
> Lorsque vous exécutez un déploiement manuel, App Runner déploie le code source à partir du référentiel complet.

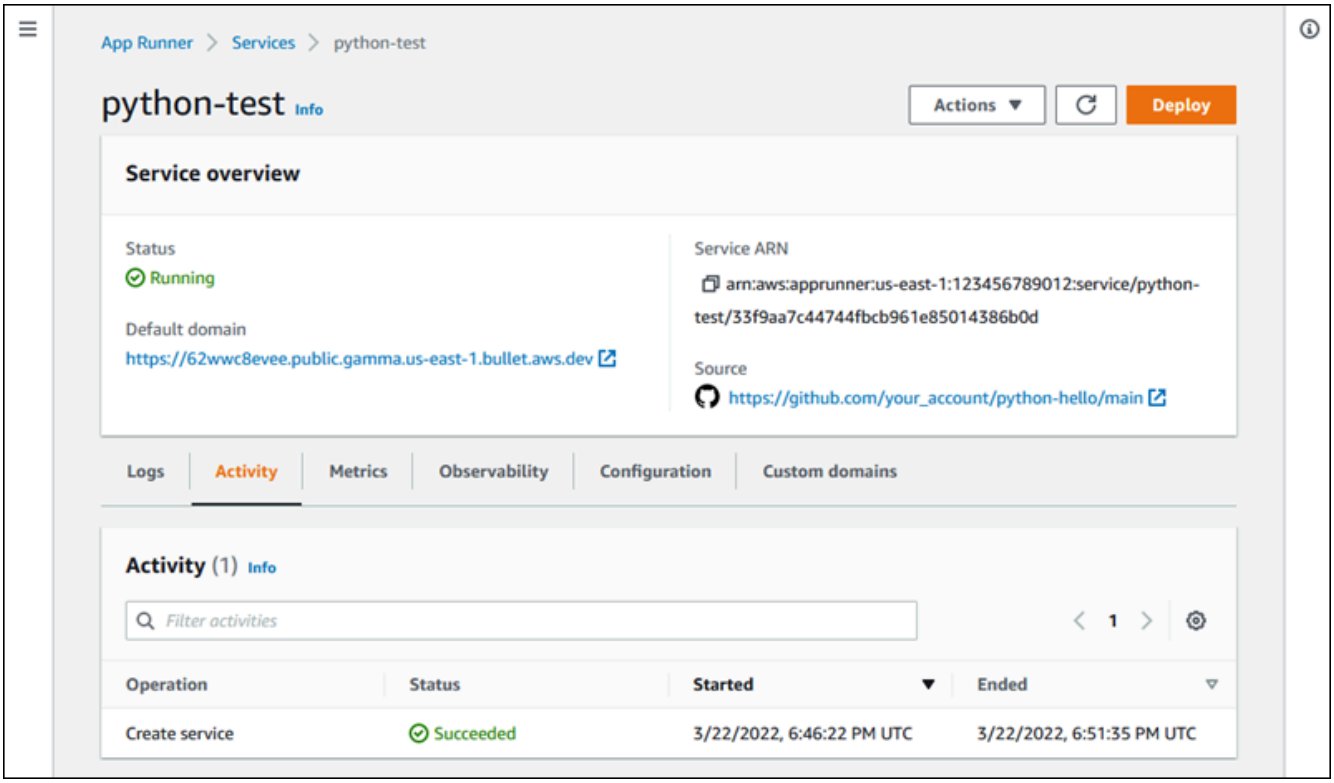
Déployez une version de votre application à l'aide de l'une des méthodes suivantes :

App Runner console

Pour déployer à l'aide de la console App Runner

1. Ouvrez la console App Runner, puis dans la liste des régions, sélectionnez votre Région AWS.
2. Dans le volet de navigation, choisissez Services, puis choisissez votre service App Runner.

   La console affiche le tableau de bord des services avec une vue d'ensemble des services.



3. Choisissez Déployer.

   Résultat : le déploiement de la nouvelle version démarre. Sur la page du tableau de bord du service, l'état du service devient Opération en cours.

4. Attendez la fin du déploiement. Sur la page du tableau de bord du service, le statut du service doit redevenir En cours d'exécution.

5. Pour vérifier que le déploiement est réussi, sur la page du tableau de bord du service, choisissez la valeur de domaine par défaut, c'est-à-dire l'URL du site Web de votre service. Inspectez ou interagissez avec votre application Web et vérifiez votre changement de version.

> **Note**
>
> Pour renforcer la sécurité de vos applications App Runner, le domaine*.awsapprunner.com est enregistré dans la liste des suffixes publics (PSL). Pour plus de sécurité, nous vous recommandons d'utiliser des cookies avec un `__Host-` préfixe si vous devez définir des cookies sensibles dans le nom de domaine par défaut de vos applications App Runner. Cette pratique vous aidera à protéger votre domaine contre les tentatives de falsification de requêtes intersites

(CSRF). Pour plus d'informations, consultez la page Set-Cookie du Mozilla Developer Network.

App Runner API or AWS CLI

Pour effectuer un déploiement à l'aide de l'API App Runner AWS CLI, ou appelez l'action StartDeploymentAPI. Le seul paramètre à transmettre est l'ARN de votre service. Vous avez déjà configuré l'emplacement de la source de votre application lorsque vous avez créé le service, et App Runner peut trouver la nouvelle version. Votre déploiement démarre si l'appel renvoie une réponse positive.

Configuration d'un service App Runner

Lorsque vous créez un AWS App Runner service, vous définissez différentes valeurs de configuration. Vous pouvez modifier certains de ces paramètres de configuration après avoir créé le service. Les autres paramètres ne peuvent être appliqués que lors de la création du service et ne peuvent pas être modifiés par la suite. Cette rubrique décrit la configuration de votre service à l'aide de l'API App Runner, de la console App Runner et d'un fichier de configuration App Runner.

Rubriques

- Configurez votre service à l'aide de l'API App Runner ou AWS CLI
- Configurez votre service à l'aide de la console App Runner
- Configurez votre service à l'aide d'un fichier de configuration App Runner
- Configuration de l'observabilité pour votre service
- Configuration des paramètres de service à l'aide de ressources partageables
- Configuration des bilans de santé pour votre service

Configurez votre service à l'aide de l'API App Runner ou AWS CLI

L'API définit les paramètres qui peuvent être modifiés après la création du service. La liste suivante décrit les actions, les types et les limites pertinents.

- UpdateServiceaction — Peut être appelée après la création pour mettre à jour certains paramètres de configuration.

- Peut être mis à jour — Vous pouvez mettre à jour les paramètres dans les `HealthCheckConfiguration` paramètres `SourceConfigurationInstanceConfiguration`, et. Cependant, dans`SourceConfiguration`, vous ne pouvez pas changer le type de source du code à l'image ou inversement. Vous devez fournir le même paramètre de référentiel que celui que vous avez indiqué lors de la création du service. C'est l'un `CodeRepository` ou l'autre`ImageRepository`.

  Vous pouvez également mettre à jour les ressources ARNs de configuration distinctes suivantes associées au service :

  - `AutoScalingConfigurationArn`
  - `VpcConnectorArn`
- Impossible de mettre à jour : vous ne pouvez pas modifier les `EncryptionConfiguration` paramètres `ServiceName` et disponibles dans l'CreateServiceaction. Ils ne peuvent pas être modifiés une fois qu'ils ont été créés. L'UpdateServiceaction n'inclut pas ces paramètres.
- API ou fichier : vous pouvez définir le `ConfigurationSource` paramètre du CodeConfigurationtype (utilisé pour les référentiels de code source dans le cadre de`SourceConfiguration`) sur`Repository`. Dans ce cas, App Runner ignore les paramètres de configuration et lit ces paramètres à partir d'un fichier de configuration de votre référentiel. `CodeConfigurationValues` Si vous définissez cette `ConfigurationSource` option`API`, App Runner obtient tous les paramètres de configuration depuis l'appel d'API et ignore le fichier de configuration, même s'il en existe un.

- TagResourceaction — Peut être appelée après la création de votre service pour ajouter des balises au service ou mettre à jour les valeurs des balises existantes.
- UntagResourceaction — Peut être appelée après la création de votre service pour supprimer des balises du service.

> **Note**
>
> Si vous créez un connecteur VPC pour le trafic sortant pour un service, le processus de démarrage du service qui suit connaîtra une latence unique. Vous pouvez définir cette configuration pour un nouveau service lorsque vous le créez, ou ultérieurement, avec une mise à jour du service. Pour plus d'informations, consultez le Latence unique chapitre de ce guide consacré à la mise en réseau avec App Runner.

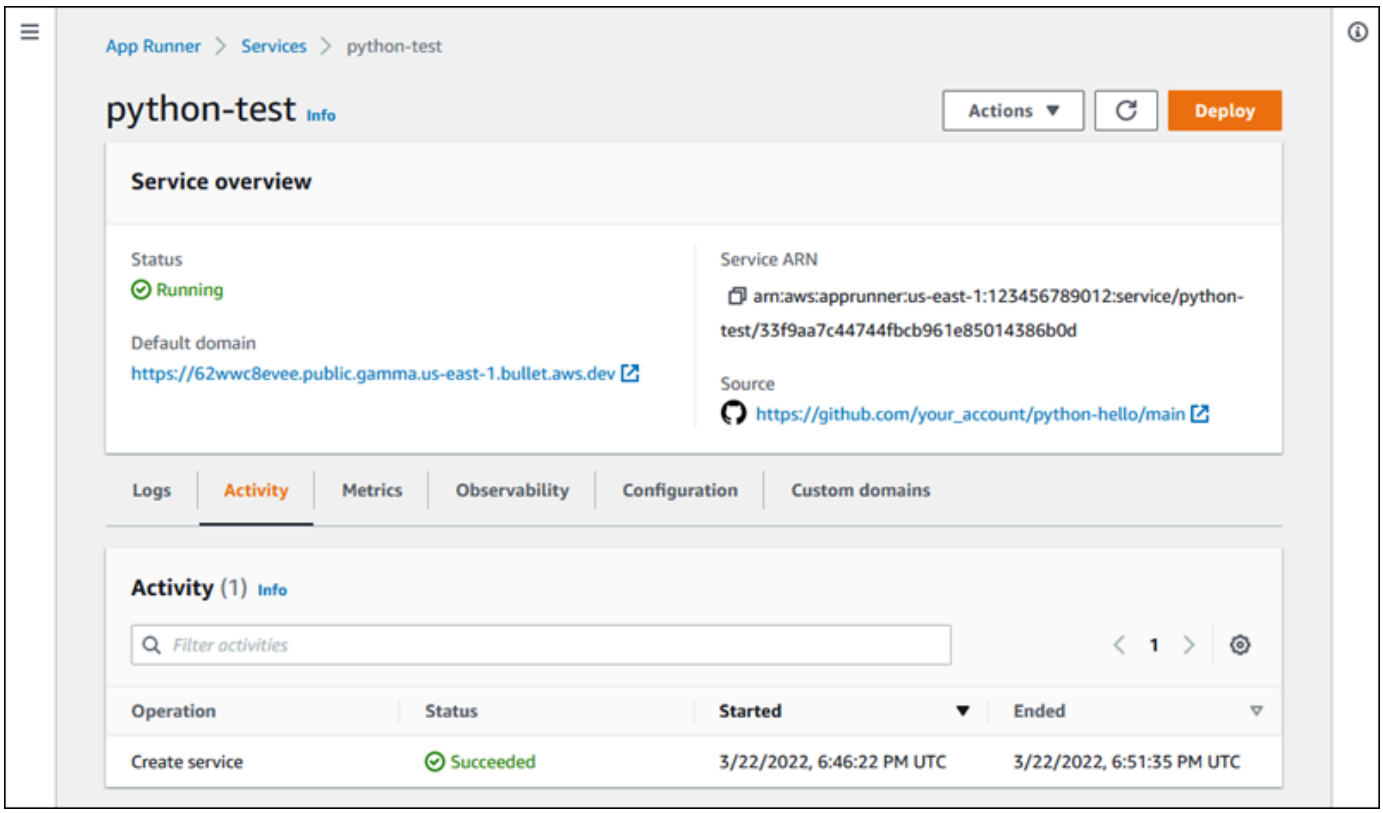
Configurez votre service à l'aide de la console App Runner

La console utilise l'API App Runner pour appliquer les mises à jour de configuration. Les règles de mise à jour imposées par l'API, telles que définies dans la section précédente, déterminent ce que vous pouvez configurer à l'aide de la console. Certains paramètres qui étaient disponibles lors de la création du service ne peuvent pas être modifiés ultérieurement. En outre, si vous décidez d'utiliser un fichier de configuration, les paramètres supplémentaires sont masqués dans la console et App Runner les lit à partir du fichier.

Pour configurer votre service

1. Ouvrez la console App Runner, puis dans la liste des régions, sélectionnez votre Région AWS.
2. Dans le volet de navigation, choisissez Services, puis choisissez votre service App Runner.

   La console affiche le tableau de bord des services avec une vue d'ensemble des services.



3. Sur la page du tableau de bord du service, choisissez l'onglet Configuration.

   Résultat : La console affiche les paramètres de configuration actuels de votre service dans plusieurs sections : Source et déploiement, Configuration du build et Configuration du service.

4. Pour mettre à jour les paramètres de n'importe quelle catégorie, choisissez Modifier.

5. Sur la page de modification de la configuration, apportez les modifications souhaitées, puis choisissez Enregistrer les modifications.

> **Note**
>
> Si vous créez un connecteur VPC pour le trafic sortant pour un service, le processus de démarrage du service qui suit connaîtra une latence unique. Vous pouvez définir cette configuration pour un nouveau service lorsque vous le créez, ou ultérieurement, avec une mise à jour du service. Pour plus d'informations, consultez le Latence unique chapitre de ce guide consacré à la mise en réseau avec App Runner.

Configurez votre service à l'aide d'un fichier de configuration App Runner

Lorsque vous créez ou mettez à jour un service App Runner, vous pouvez demander à App Runner de lire certains paramètres de configuration à partir d'un fichier de configuration que vous fournissez dans le cadre de votre référentiel source. Ce faisant, vous pouvez gérer les paramètres liés à votre code source sous contrôle de source, ainsi que le code lui-même. Le fichier de configuration fournit également certains paramètres avancés que vous ne pouvez pas définir à l'aide de la console ou de l'API. Pour de plus amples informations, veuillez consulter Fichier de configuration d'App Runner.

> **Note**
>
> Si vous créez un connecteur VPC pour le trafic sortant pour un service, le processus de démarrage du service qui suit connaîtra une latence unique. Vous pouvez définir cette configuration pour un nouveau service lorsque vous le créez, ou ultérieurement, avec une mise à jour du service. Pour plus d'informations, consultez le Latence unique chapitre de ce guide consacré à la mise en réseau avec App Runner.

Configuration de l'observabilité pour votre service

AWS App Runner s'intègre à plusieurs AWS services afin de vous fournir une suite complète d'outils d'observabilité pour votre service App Runner. Pour de plus amples informations, veuillez consulter Observabilité.

App Runner permet d'activer certaines fonctionnalités d'observabilité et de configurer leur comportement à l'aide d'une ressource partageable appelée. ObservabilityConfiguration Vous pouvez

fournir une ressource de configuration d'observabilité lorsque vous créez ou mettez à jour un service. La console App Runner en crée un pour vous lorsque vous créez un nouveau service App Runner. La fourniture d'une configuration d'observabilité est facultative. Si vous n'en fournissez pas, App Runner fournit une configuration d'observabilité par défaut.

Vous pouvez partager une configuration d'observabilité unique entre plusieurs services App Runner afin de vous assurer qu'ils ont le même comportement d'observabilité. Pour de plus amples informations, veuillez consulter the section called “Ressources de configuration”.

Vous pouvez configurer les fonctionnalités d'observabilité suivantes à l'aide des configurations d'observabilité :

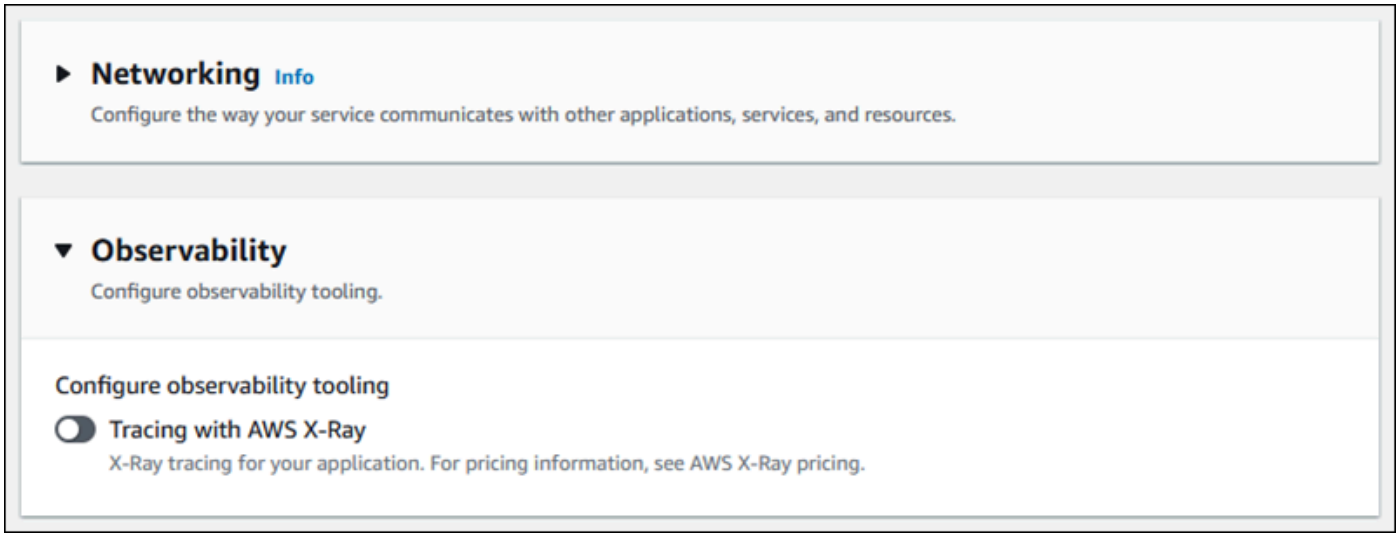
- Configuration du suivi : paramètres de suivi des demandes traitées par votre application et des appels en aval qu'elle effectue. Pour plus d’informations sur le suivi, consultez the section called “Traçage (X-Ray)”.

Gérez l'observabilité

Gérez l'observabilité de vos services App Runner à l'aide de l'une des méthodes suivantes :

App Runner console

Lorsque vous créez un service à l'aide de la console App Runner ou que vous mettez à jour sa configuration ultérieurement, vous pouvez configurer les fonctionnalités d'observabilité de votre service. Consultez la section de configuration de l'observabilité sur la page de la console.



App Runner API or AWS CLI

Lorsque vous appelez les actions CreateServiceou UpdateServicel'API App Runner, vous pouvez utiliser l'objet `ObservabilityConfiguration` paramètre pour activer les fonctionnalités d'observabilité et spécifier une ressource de configuration d'observabilité pour votre service.

Utilisez les actions d'API App Runner suivantes pour gérer vos ressources de configuration d'observabilité.

- CreateObservabilityConfiguration— Crée une nouvelle configuration d'observabilité ou une révision d'une configuration existante.
- ListObservabilityConfigurations— Renvoie la liste des configurations d'observabilité qui vous sont associées Compte AWS, avec des informations récapitulatives.
- DescribeObservabilityConfiguration— Renvoie une description complète d'une configuration d'observabilité.
- DeleteObservabilityConfiguration— Supprime une configuration d'observabilité. Vous pouvez supprimer une révision spécifique ou la dernière révision active. Vous devrez peut-être supprimer des configurations d'observabilité inutiles si vous atteignez le quota de configuration d'observabilité pour votre. Compte AWS

Configuration des paramètres de service à l'aide de ressources partageables

Pour certaines fonctionnalités, il est judicieux de partager la configuration entre les AWS App Runner services. Par exemple, vous souhaiterez peut-être qu'un ensemble de services présente le même comportement de mise à l'échelle automatique. Il se peut également que vous souhaitiez des paramètres d'observabilité identiques pour tous vos services. App Runner vous permet de partager des paramètres en utilisant des ressources partageables distinctes. Vous créez une ressource qui définit un ensemble de paramètres de configuration pour une fonctionnalité, puis vous fournissez le nom de ressource Amazon (ARN) de cette ressource de configuration à un ou plusieurs services App Runner.

App Runner implémente des ressources de configuration partageables pour les fonctionnalités suivantes :

- Auto scaling (Mise à l'échelle automatique)
- Observabilité

- Accès VPC

La page du document correspondant à chacune de ces fonctionnalités fournit des informations sur les paramètres disponibles et les procédures de gestion.

Les fonctionnalités utilisant des ressources de configuration distinctes partagent certaines caractéristiques et considérations de conception.

- Révisions — Certaines ressources de configuration peuvent comporter des révisions. Le dimensionnement automatique et l'observabilité sont des exemples de deux ressources de configuration qui utilisent des révisions. Dans ces cas, chaque configuration possède un nom et une révision numérique. Plusieurs révisions d'une configuration portent le même nom et portent des numéros de révision différents. Vous pouvez utiliser différents noms de configuration pour différents scénarios. Pour chaque nom, vous pouvez ajouter plusieurs révisions afin d'affiner les paramètres d'un scénario spécifique.

  La première configuration que vous créez avec un nom reçoit le numéro de révision 1. Les configurations suivantes portant le même nom reçoivent des numéros de révision consécutifs (en commençant par 2). Vous pouvez associer votre service App Runner à une révision de configuration spécifique ou à la dernière révision de configuration.
- Partagé : vous pouvez partager une seule ressource de configuration entre plusieurs services App Runner. Cela est utile si vous souhaitez conserver des configurations identiques sur l'ensemble de ces services. En particulier, si vos ressources prennent en charge les révisions, vous pouvez configurer plusieurs services pour utiliser la dernière révision d'une configuration. Vous pouvez le faire en spécifiant uniquement le nom de la configuration, mais pas de révision. Tous les services que vous avez configurés de cette manière reçoivent des mises à jour de configuration lorsque vous mettez à jour le service. Pour plus d'informations sur les modifications de configuration, consultezthe section called “Configuration”.
- Gestion des ressources : vous pouvez utiliser App Runner pour créer et supprimer des configurations. Vous ne pouvez pas mettre à jour directement une configuration. En revanche, pour les ressources qui prennent en charge les révisions, vous pouvez créer une nouvelle révision d'un nom de configuration existant afin de mettre à jour efficacement la configuration.

> **Note**
>
> Pour le dimensionnement automatique, vous pouvez créer des configurations et de multiples révisions à l'aide de la console App Runner et de l'API App Runner. La console

App Runner et l'API App Runner peuvent également supprimer des configurations et des révisions. Pour en savoir plus, consultez Gestion du dimensionnement automatique d'App Runner.
Pour les autres types de configuration, tels que les configurations d'observabilité, vous ne pouvez créer une configuration avec une seule révision qu'avec la console App Runner. Pour créer d'autres révisions et supprimer des configurations, vous devez utiliser l'API App Runner.

- Quota de ressources : il existe des quotas définis pour le nombre de noms de configuration uniques et de révisions que vous pouvez avoir pour vos ressources de configuration dans chacun d'eux Région AWS. Si vous atteignez ces quotas, vous devez supprimer un nom de configuration ou au moins certaines de ses révisions avant de pouvoir en créer d'autres. Pour les révisions des configurations de dimensionnement automatique, vous pouvez utiliser la console App Runner ou l'API App Runner pour les supprimer. Pour en savoir plus, consultez Gestion du dimensionnement automatique d'App Runner. Vous devez utiliser l'API App Runner pour supprimer d'autres ressources. Pour plus d'informations sur les quotas, consultez the section called "Quotas de ressources App Runner".
- Aucun coût de ressource : la création d'une ressource de configuration n'entraîne aucun coût supplémentaire. La fonctionnalité elle-même peut vous être facturée (par exemple, le AWS X-Ray coût normal vous est facturé lorsque vous activez le suivi X-Ray), mais pas pour la ressource de configuration App Runner qui configure la fonctionnalité pour votre service App Runner.

Configuration des bilans de santé pour votre service

AWS App Runner surveille l'état de santé de votre service en effectuant des bilans de santé. Le protocole de contrôle de santé par défaut est TCP. App Runner envoie un ping au domaine attribué à votre service. Vous pouvez également définir le protocole de contrôle de santé sur HTTP. App Runner envoie des requêtes HTTP de vérification de l'état à votre application Web.

Vous pouvez configurer quelques paramètres relatifs aux bilans de santé. Le tableau suivant décrit les paramètres du bilan de santé et leurs valeurs par défaut.

Réglage	Description	Par défaut
Protocole	Le protocole IP utilisé par App Runner pour effectuer des surveillances de l'état pour votre service.	`TCP`

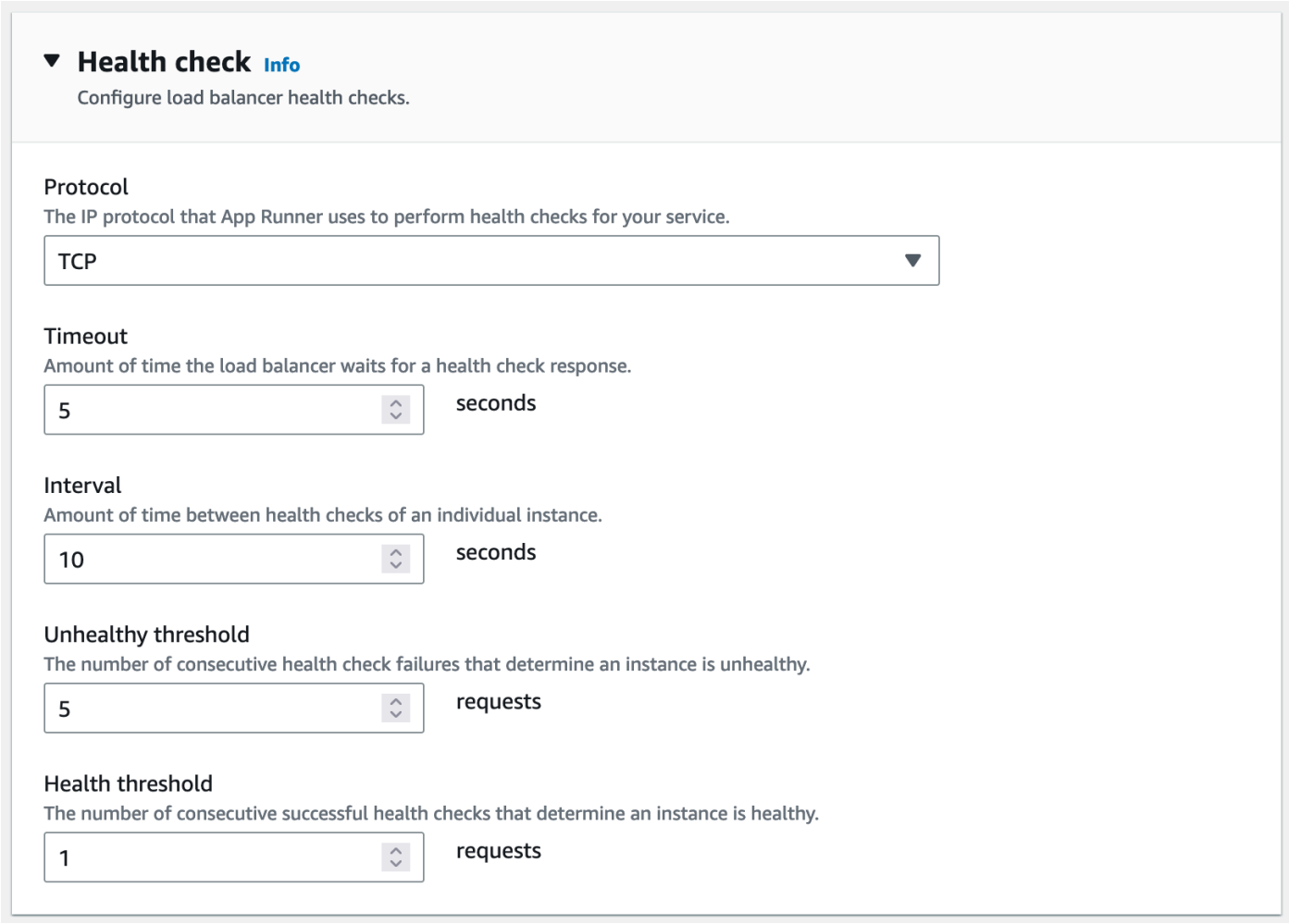
Réglage	Description	Par défaut
	Si vous définissez le protocole sur`TCP`, App Runner envoie un ping au domaine par défaut attribué à votre service sur le port que votre application écoute. Si vous définissez le protocole sur`HTTP`, App Runner envoie les demandes de contrôle de santé au chemin configuré.	
Chemin	URL à laquelle App Runner envoie les demandes de contrôle de santé HTTP. Applicable uniquement aux vérifications HTTP.	`/`
Interval	L'intervalle, en secondes, entre les surveillances de l'état.	5
Timeout	Le temps d'attente, en secondes, d'une réponse de surveillance de l'état avant de décider qu'elle a échoué.	2
Seuil sain	Nombre de contrôles consécutifs qui doivent réussir avant qu'App Runner ne décide que le service est sain.	1
Seuil malsain	Le nombre de contrôles consécutifs qui doivent échouer avant qu'App Runner ne décide que le service n'est pas sain.	5

Configurer la surveillance de l'état

Configurez les contrôles de santé de votre service App Runner à l'aide de l'une des méthodes suivantes :

App Runner console

Lorsque vous créez votre service App Runner à l'aide de la console App Runner, ou lorsque vous mettez à jour sa configuration ultérieurement, vous pouvez configurer les paramètres de contrôle de santé. Pour les procédures complètes relatives à la console, reportez-vous the section called “Création” aux sections etthe section called “Configuration”. Dans les deux cas, consultez la section de configuration du Health check sur la page de la console.



App Runner API or AWS CLI

Lorsque vous appelez les actions de l'UpdateServiceAPI CreateServiceor, vous pouvez utiliser le `HealthCheckConfiguration` paramètre pour spécifier les paramètres de contrôle de santé.

Pour plus d'informations sur la structure du paramètre, reportez-vous HealthCheckConfigurationà la référence de l'AWS App Runner API.

Gestion des connexions App Runner

Lorsque vous créez un service dans AWS App Runner, vous configurez une source d'application : une image de conteneur ou un référentiel source stocké auprès d'un fournisseur. App Runner doit établir une connexion authentifiée et autorisée avec le fournisseur. App Runner peut ensuite lire votre référentiel et le déployer sur votre service. App Runner ne nécessite pas d'établissement de connexion lorsque vous créez un service qui accède au code stocké dans votre Compte AWS.

App Runner conserve les informations de connexion dans une ressource appelée connexion. La console App Runner et ce guide qualifient également les connexions de comptes connectés. App Runner nécessite une ressource de connexion lorsque vous créez un service nécessitant des informations de connexion tierces. Voici quelques informations importantes concernant les connexions :

- Fournisseurs — App Runner nécessite actuellement des ressources de connexion avec GitHubou Bitbucket.
- Partagé : vous pouvez utiliser une ressource de connexion pour créer plusieurs services App Runner utilisant le même compte de fournisseur de référentiel.
- Gestion des ressources : dans App Runner, vous pouvez créer et supprimer des connexions. Cependant, vous ne pouvez pas modifier une connexion existante.
- Quota de ressources : les ressources de connexion ont un quota défini qui est associé au vôtre Compte AWS dans chacune d'entre elles Région AWS. Si vous atteignez ce quota, vous devrez peut-être supprimer une connexion avant de pouvoir vous connecter à un nouveau compte fournisseur. Vous pouvez supprimer une connexion à l'aide de la console ou de l'API App Runner, comme décrit dans la section suivantethe section called “Gérer les connexions”. Pour de plus amples informations, veuillez consulter the section called “Quotas de ressources App Runner”.

Gérer les connexions

Gérez vos connexions App Runner à l'aide de l'une des méthodes suivantes :

App Runner console

Lorsque vous utilisez la console App Runner pour créer un service, vous fournissez les détails de connexion. Il n'est pas nécessaire de créer explicitement une ressource de connexion. Dans la console, vous pouvez choisir de vous connecter à un GitHub compte Bitbucket auquel vous vous êtes déjà connecté, ou de vous connecter à un nouveau compte. Si nécessaire, App Runner crée une ressource de connexion pour vous. Pour une nouvelle connexion, certains fournisseurs exigent que vous effectuiez une poignée de main d'authentification avant de pouvoir utiliser la connexion. La console vous guide tout au long de ce processus.

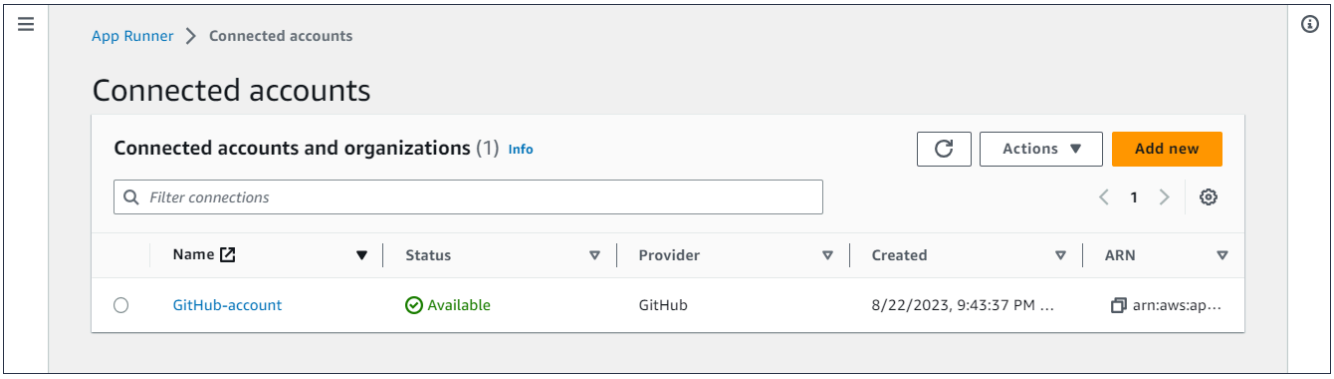
La console comporte également une page permettant de gérer vos connexions existantes. Vous pouvez terminer le handshake d'authentification pour une connexion si vous ne l'avez pas fait lors de la création de votre service. Vous pouvez également supprimer les connexions que vous

n'utilisez plus. La procédure suivante montre comment gérer les connexions aux fournisseurs de référentiels.

Pour gérer les connexions dans votre compte

1. Ouvrez la console App Runner, puis dans la liste des régions, sélectionnez votre Région AWS.
2. Dans le volet de navigation, sélectionnez Comptes connectés.

   La console affiche ensuite une liste des connexions aux fournisseurs de référentiels dans votre compte.



3. Vous pouvez désormais effectuer l'une des actions suivantes avec n'importe quelle connexion de la liste :

   - Ouvrir GitHub/Bitbucket un compte ou une organisation : choisissez le nom de la connexion.
   - Poignée de main d'authentification complète : sélectionnez la connexion, puis dans le menu Actions, choisissez Terminer la poignée de main. La console vous guide tout au long du processus d'authentification.
   - Supprimer la connexion : sélectionnez la connexion, puis dans le menu Actions, choisissez Supprimer. Suivez les instructions de l'invite de suppression.

App Runner API or AWS CLI

Vous pouvez utiliser les actions d'API App Runner suivantes pour gérer vos connexions.

- CreateConnection— Crée une connexion avec un compte de fournisseur de référentiel. Une fois la connexion créée, vous devez effectuer manuellement le handshake d'authentification à l'aide de la console App Runner. Ce processus est expliqué dans la section précédente.

- ListConnections— Renvoie la liste des connexions App Runner associées à votre Compte AWS.
- DeleteConnection— Supprime une connexion. Vous devrez peut-être supprimer les connexions inutiles si vous atteignez le quota de connexions pour votre Compte AWS.

Gestion du dimensionnement automatique d'App Runner

AWS App Runner redimensionne automatiquement les ressources de calcul, en particulier les instances, à la hausse ou à la baisse pour votre application App Runner. La mise à l'échelle automatique permet une gestion adéquate des demandes lorsque le trafic est dense et réduit les coûts lorsque le trafic ralentit.

Configuration de mise à l'échelle automatique

Vous pouvez configurer quelques paramètres pour ajuster le comportement de dimensionnement automatique de votre service. App Runner gère les paramètres de mise à l'échelle automatique dans une ressource partageable appelée AutoScalingConfiguration. Vous pouvez créer et gérer des configurations de mise à l'échelle automatique autonomes avant de les attribuer à des services. Une fois qu'ils ont été associés à un service, vous pouvez continuer à gérer les configurations. Vous pouvez également choisir de créer une nouvelle configuration de dimensionnement automatique pendant que vous créez un nouveau service ou que vous configurez un service existant. Une fois la nouvelle configuration de dimensionnement automatique créée, vous pouvez l'associer au service et poursuivre le processus de création ou de configuration de votre service.

Dénomination et révisions

Une configuration de mise à l'échelle automatique possède un nom et une révision numérique. Plusieurs révisions d'une configuration portent le même nom et portent des numéros de révision différents. Vous pouvez utiliser différents noms de configuration pour différents scénarios de mise à l'échelle automatique, tels que la haute disponibilité ou le faible coût. Pour chaque nom, vous pouvez ajouter plusieurs révisions afin d'affiner les paramètres d'un scénario spécifique. Vous pouvez avoir jusqu'à 10 noms de configuration d'autoscaling uniques et jusqu'à 5 révisions pour chaque configuration. Si vous atteignez la limite et que vous devez en créer d'autres, vous pouvez en supprimer une, puis en créer une autre. App Runner ne vous permettra pas de supprimer une configuration définie par défaut ou utilisée par un service actif. Pour plus d’informations sur les quotas, consultez the section called “Quotas de ressources App Runner”.

Définition d'une configuration par défaut

Lorsque vous créez ou mettez à jour un service App Runner, vous pouvez fournir une ressource de configuration à dimensionnement automatique. La fourniture d'une configuration de mise à l'échelle automatique est facultative. Si vous n'en fournissez pas, App Runner fournit une configuration de dimensionnement automatique par défaut avec des valeurs recommandées. La fonction de configuration de mise à l'échelle automatique vous permet de définir votre propre configuration de mise à l'échelle automatique par défaut au lieu d'utiliser la configuration par défaut fournie par App Runner. Une fois que vous avez défini une autre configuration de dimensionnement automatique par défaut, cette configuration est automatiquement attribuée par défaut aux nouveaux services que vous créerez à l'avenir. La nouvelle désignation par défaut n'affecte pas les associations précédemment définies pour les services existants.

Configuration des services avec mise à l'échelle automatique

Vous pouvez partager une configuration de dimensionnement automatique unique entre plusieurs services App Runner afin de garantir que les services ont le même comportement de dimensionnement automatique. Pour plus d'informations sur la configuration des configurations de dimensionnement automatique à l'aide de la console App Runner ou de l'API App Runner, consultez les sections suivantes de cette rubrique. Pour des informations plus générales sur les ressources partageables, consultezthe section called “Ressources de configuration”.

Réglages configurables

Vous pouvez configurer les paramètres de mise à l'échelle automatique suivants :

- Concurrence maximale : nombre maximal de demandes simultanées traitées par une instance. Lorsque le nombre de demandes simultanées dépasse ce quota, App Runner augmente le service.
- Taille maximale : nombre maximal d'instances que votre service peut atteindre. Il s'agit du plus grand nombre d'instances pouvant gérer simultanément le trafic de votre service.
- Taille minimale : nombre minimum d'instances qu'App Runner peut fournir pour votre service. Le service dispose toujours d'au moins ce nombre d'instances provisionnées. Certaines de ces instances gèrent activement le trafic. Les autres font partie de la réserve de capacité de calcul rentable, qui est prête à être rapidement activée. Vous payez pour l'utilisation de la mémoire de toutes les instances provisionnées. Vous ne payez que pour l'utilisation du processeur du sous-ensemble actif.

> **Note**
>
> Le nombre de ressources vCPU détermine le nombre d'instances qu'App Runner peut fournir à votre service. Il s'agit d'une valeur de quota ajustable pour le nombre de ressources de vCPU Fargate On-Demand résidant dans le service. AWS Fargate Pour consulter les paramètres de quota de vCPU de votre compte ou pour demander une augmentation de quota, utilisez la console Service Quotas dans le. AWS Management Console Pour plus d'informations, consultez les quotas AWS Fargate de service dans le manuel Amazon Elastic Container Service Developer Guide.

Gérer le dimensionnement automatique d'un service

Gérez le dimensionnement automatique de vos services App Runner à l'aide de l'une des méthodes suivantes :

App Runner console

Lorsque vous créez un service à l'aide de la console App Runner ou que vous mettez à jour une configuration de service, vous pouvez spécifier une configuration de dimensionnement automatique.

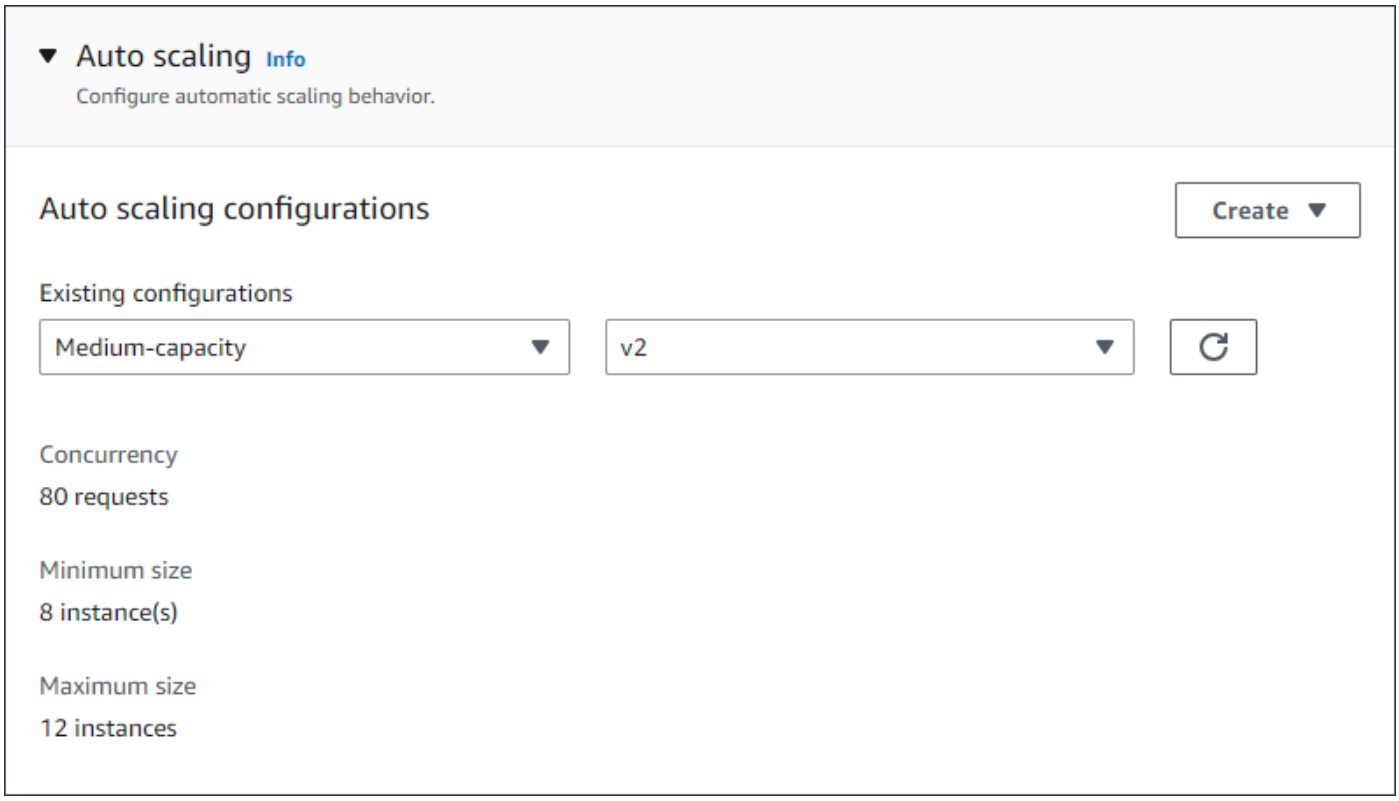
> **Note**
>
> Lorsque vous modifiez la configuration ou la révision du dimensionnement automatique associée à un service, celui-ci est redéployé.

La page de configuration du dimensionnement automatique propose plusieurs options pour configurer le dimensionnement automatique de votre service.

- Pour attribuer une configuration et une révision existantes : choisissez une valeur dans le menu déroulant Configurations existantes. La dernière version de révision sera affichée par défaut dans la liste déroulante adjacente. S'il existe une autre révision que vous préférez sélectionner, faites-le dans le menu déroulant des révisions. Les valeurs de configuration pour la version de révision s'affichent.
- Pour créer et attribuer une nouvelle configuration de dimensionnement automatique, sélectionnez Créer un nouvel ASC dans le menu Créer. Cela ouvre la page de configuration

Ajouter une mise à l'échelle automatique personnalisée. Entrez un nom de configuration et des valeurs pour les paramètres de mise à l'échelle automatique. Sélectionnez ensuite Ajouter. App Runner crée pour vous la nouvelle ressource de configuration de mise à l'échelle automatique et vous renvoie à la section de mise à l'échelle automatique avec la nouvelle configuration sélectionnée et affichée.

- Pour créer et attribuer une nouvelle révision : sélectionnez d'abord le nom de la configuration dans le menu déroulant Configurations existantes. Sélectionnez ensuite Créer une révision ASC dans le menu Créer. Cela ouvre la page de configuration Ajouter une mise à l'échelle automatique personnalisée. Entrez des valeurs pour les paramètres de mise à l'échelle automatique. Sélectionnez ensuite Ajouter. App Runner crée pour vous une nouvelle révision de configuration de mise à l'échelle automatique et vous renvoie à la section de mise à l'échelle automatique avec la nouvelle révision sélectionnée et affichée.



App Runner API or AWS CLI

Lorsque vous appelez les actions CreateServiceou UpdateServicel'API App Runner, vous pouvez utiliser le `AutoScalingConfigurationArn` paramètre pour spécifier une ressource de configuration à dimensionnement automatique pour votre service.

La section suivante fournit des conseils pour gérer vos ressources de configuration d'autoscaling.

Gérez les ressources de configuration de mise à l'échelle automatique

Gérez les configurations et les révisions du dimensionnement automatique d'App Runner pour votre compte à l'aide de l'une des méthodes suivantes :

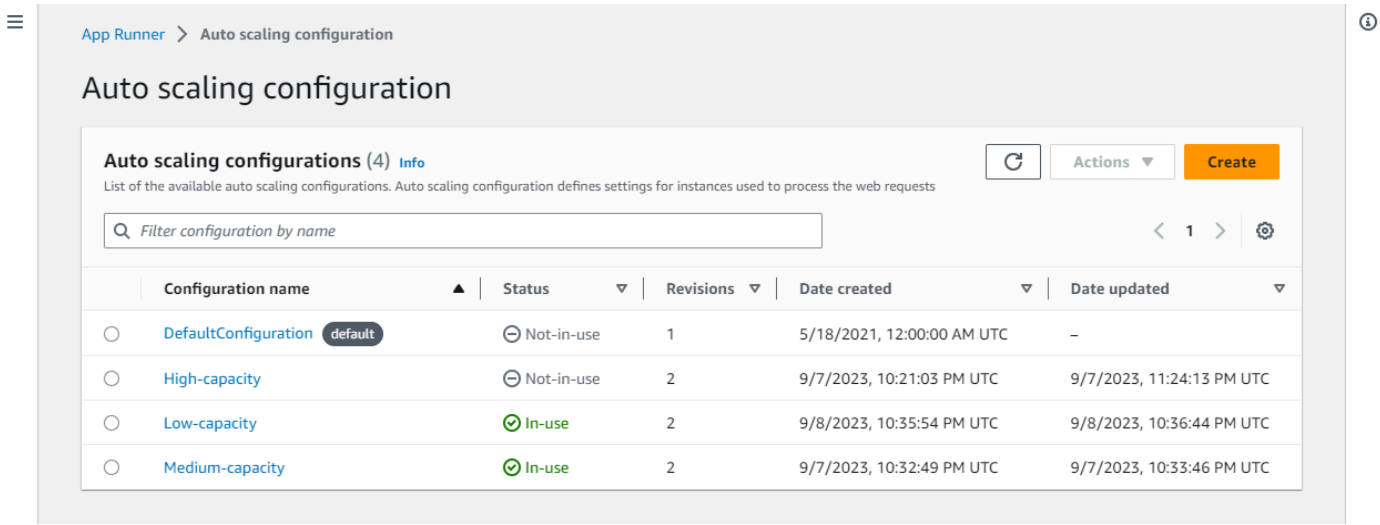
App Runner console

Gérez les configurations de mise à l'échelle automatique

La page Configurations de dimensionnement automatique répertorie les configurations de dimensionnement automatique que vous avez configurées dans votre compte. Vous pouvez créer et gérer vos configurations d'autoscaling sur cette page, puis les attribuer ultérieurement à un ou plusieurs services App Runner.

Vous pouvez effectuer l'une des opérations suivantes à partir de cette page :

- Créez une nouvelle configuration de mise à l'échelle automatique.
- Créez une nouvelle révision pour une configuration auto scaling existante.
- Supprimez une configuration de mise à l'échelle automatique.
- Définissez une configuration de mise à l'échelle automatique par défaut.



Pour gérer les configurations de mise à l'échelle automatique dans votre compte

1. Ouvrez la console App Runner, puis dans la liste des régions, sélectionnez votre Région AWS.

2. Dans le volet de navigation, sélectionnez Configurations de mise à l'échelle automatique. La console affiche la liste des configurations de dimensionnement automatique de votre compte.

   Vous pouvez désormais effectuer l'une des opérations suivantes.

   - Pour créer une nouvelle configuration de mise à l'échelle automatique, procédez comme suit.

     a. Sur la page Configurations du dimensionnement automatique, sélectionnez Créer.

        La page Créer une configuration de dimensionnement automatique s'affiche.

     b. Entrez des valeurs pour le nom de la configuration, la simultanéité, la taille minimale et la taille maximale.

     c. (Facultatif) Si vous souhaitez ajouter des balises, sélectionnez Nouvelle balise automatique. Ensuite, dans les champs qui apparaissent, entrez un nom et une valeur (facultatif).

     d. Sélectionnez Créer.

   - Pour créer une nouvelle révision pour une configuration auto scaling existante, procédez comme suit.

     a. Sur la page Configurations du dimensionnement automatique, sélectionnez le bouton radio à côté de la configuration qui nécessite la nouvelle révision. Sélectionnez ensuite Créer une révision dans le menu Actions.

        La page Créer une révision s'affiche.

     b. Activé, entrez des valeurs pour la simultanéité, la taille minimale et la taille maximale.

     c. (Facultatif) Si vous souhaitez ajouter des balises, sélectionnez Nouvelle balise automatique. Ensuite, dans les champs qui apparaissent, entrez un nom et une valeur (facultatif).

     d. Sélectionnez Créer.

   - Pour supprimer une configuration de mise à l'échelle automatique, procédez comme suit.

     a. Sur la page Configurations du dimensionnement automatique, sélectionnez le bouton radio à côté de la configuration que vous devez supprimer.

     b. Sélectionnez Supprimer dans le menu Actions.

c. Pour procéder à la suppression, sélectionnez Supprimer dans la boîte de dialogue de confirmation. Sinon, sélectionnez Annuler.

> **Note**
>
> App Runner vérifie que votre choix de suppression n'est pas défini par défaut ou qu'il est actuellement utilisé par un service actif.

- Pour définir une configuration de dimensionnement automatique comme configuration par défaut, procédez comme suit.

  a. Sur la page Configurations du dimensionnement automatique, sélectionnez le bouton radio à côté de la configuration que vous devez définir par défaut.

  b. Sélectionnez Définir par défaut dans le menu Actions.

  c. Une boîte de dialogue s'affiche pour vous informer qu'App Runner utilisera la dernière version comme configuration par défaut pour tous les nouveaux services que vous créez. Sélectionnez Confirmer pour continuer. Sinon, sélectionnez Annuler.

  > **Note**
  >
  > - Lorsque vous définissez une configuration de dimensionnement automatique par défaut, elle est automatiquement attribuée en tant que configuration par défaut aux nouveaux services que vous créerez à l'avenir.
  > - La nouvelle désignation par défaut n'affecte pas les associations précédemment définies pour les services existants.
  > - Si la configuration de mise à l'échelle automatique par défaut désignée comporte des révisions, App Runner attribue sa dernière révision comme version par défaut.

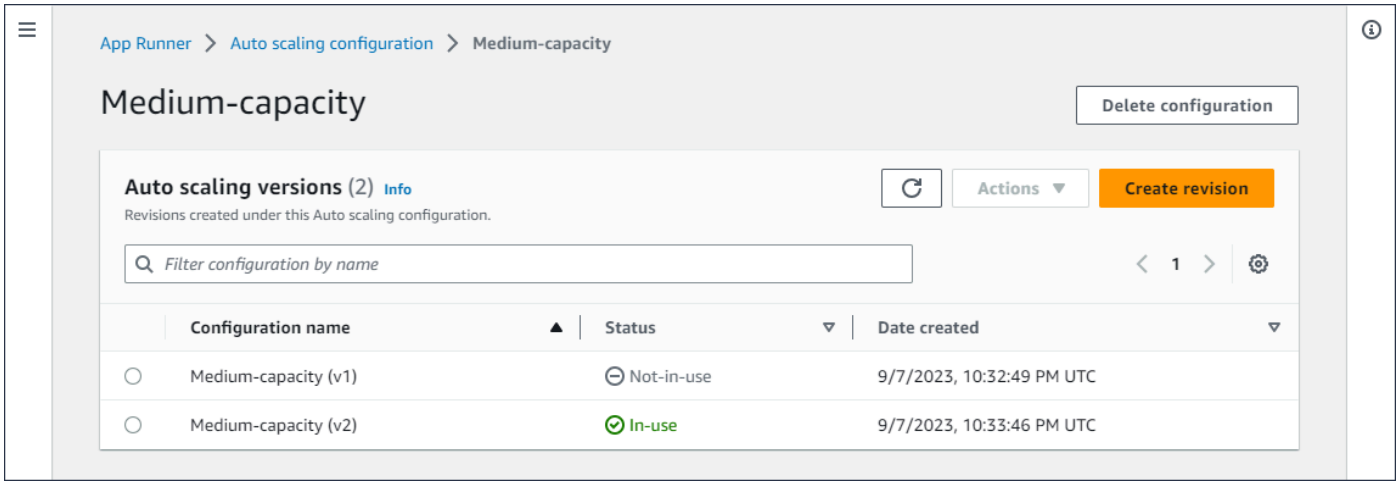
Gérer les révisions

La console dispose également d'une page permettant de créer et de gérer vos révisions de mise à l'échelle automatique existantes, appelée révisions de mise à l'échelle automatique. Accédez

à cette page en sélectionnant le nom d'une configuration sur la page des configurations Auto Scaling.

Vous pouvez effectuer l'une des opérations suivantes à partir de la page Révisions du dimensionnement automatique :

- Créez une nouvelle révision de mise à l'échelle automatique.
- Définissez une révision de configuration de mise à l'échelle automatique par défaut.
- Supprimez une révision.
- Supprimez l'intégralité de la configuration de mise à l'échelle automatique, y compris toutes les révisions associées.
- Consultez les détails de configuration d'une révision.
- Afficher la liste des services associés à une révision.
- Modifiez la révision d'un service répertorié.



Pour gérer les révisions de mise à l'échelle automatique dans votre compte

1. Ouvrez la console App Runner, puis dans la liste des régions, sélectionnez votre Région AWS.
2. Dans le volet de navigation, sélectionnez Configurations de mise à l'échelle automatique. La console affiche la liste des configurations de dimensionnement automatique de votre compte. L'ensemble de procédures précédent de cette Gérez les configurations de mise à l'échelle automatique section inclut une image d'écran de cette page.
3. Vous pouvez désormais accéder à une configuration de mise à l'échelle automatique spécifique pour visualiser et gérer toutes ses révisions. Dans le volet Configurations de mise

à l'échelle automatique, sous la colonne Nom de la configuration, choisissez un nom de configuration de mise à l'échelle automatique. Sélectionnez le nom réel plutôt que le bouton radio. Cela vous dirigera vers une liste de toutes les révisions de cette configuration sur la page des révisions de Auto Scaling.

4. Vous pouvez désormais effectuer l'une des opérations suivantes.

   - Pour créer une nouvelle révision pour une configuration auto scaling existante, procédez comme suit.

     a. Sur la page Révisions du dimensionnement automatique, sélectionnez Créer une révision.

        La page Créer une révision s'affiche.

     b. Entrez des valeurs pour la simultanéité, la taille minimale et la taille maximale.

     c. (Facultatif) Si vous souhaitez ajouter des balises, sélectionnez Nouvelle balise automatique. Ensuite, dans les champs qui apparaissent, entrez un nom et une valeur (facultatif).

     d. Sélectionnez Créer.

   - Pour supprimer l'intégralité de la configuration de mise à l'échelle automatique, y compris toutes les révisions associées, procédez comme suit.

     a. Sélectionnez Supprimer la configuration en haut à droite de la page.

     b. Pour procéder à la suppression, sélectionnez Supprimer dans la boîte de dialogue de confirmation. Sinon, sélectionnez Annuler.

     > **Note**
     >
     > App Runner vérifie que votre choix de suppression n'est pas défini par défaut ou qu'il est actuellement utilisé par un service actif.

   - Pour définir une révision de mise à l'échelle automatique par défaut, procédez comme suit.

     a. Sélectionnez le bouton radio à côté de la révision que vous devez définir comme version par défaut.

     b. Sélectionnez Définir par défaut dans le menu Actions.

> **Note**
>
> - Lorsque vous définissez une configuration de dimensionnement automatique par défaut, elle est automatiquement attribuée en tant que configuration par défaut aux nouveaux services que vous créerez à l'avenir.
> - La nouvelle désignation par défaut n'affecte pas les associations précédemment définies pour les services existants.

- Pour consulter les détails de configuration d'une révision, procédez comme suit.
  - Sélectionnez le bouton radio à côté de la révision.

    Les détails de configuration de la révision, y compris l'ARN, s'affichent dans le panneau divisé inférieur. Reportez-vous à l'image d'écran à la fin de cette procédure.
- Pour consulter la liste des services associés à une révision, procédez comme suit.
  - Sélectionnez le bouton radio à côté de la révision.

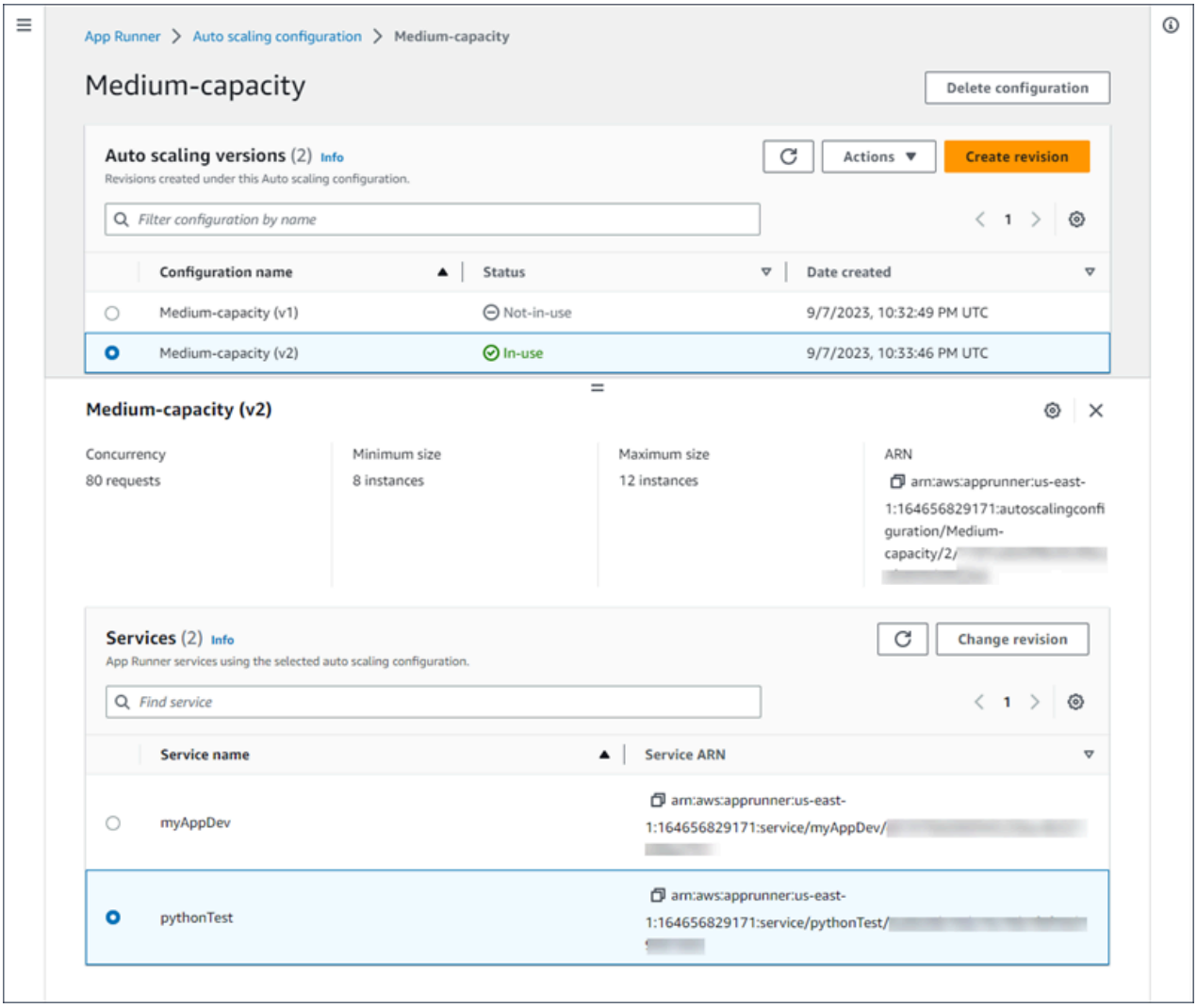
    Le panneau Services s'affiche dans le panneau divisé inférieur, sous les détails de configuration des révisions. Le panneau répertorie tous les services qui utilisent cette révision de configuration de mise à l'échelle automatique. Reportez-vous à l'image d'écran à la fin de cette procédure.
- Pour modifier la révision d'un service répertorié, procédez comme suit.
  a. Sélectionnez le bouton radio à côté de la révision, si ce n'est pas déjà fait.

     Le panneau Services s'affiche dans le panneau divisé inférieur, sous les détails de configuration des révisions. Le panneau répertorie tous les services qui utilisent cette révision de configuration de mise à l'échelle automatique. Reportez-vous à l'image d'écran à la fin de cette procédure.
  b. Dans le panneau Services, sélectionnez le bouton radio à côté du service que vous souhaitez modifier. Sélectionnez ensuite Modifier les révisions.
  c. Le panneau Change ASC revision s'affiche. Choisissez parmi les révisions disponibles dans le menu déroulant. Seules les révisions de la configuration de mise

à l'échelle automatique que vous avez choisie précédemment sont disponibles. Si vous devez passer à une autre configuration de mise à l'échelle automatique, suivez les procédures décrites dans la section précédentethe section called “Gérer le dimensionnement automatique d'un service”.

Sélectionnez Mettre à jour pour procéder à la modification. Sinon, sélectionnez Annuler.

> **Note**
>
> Lorsque vous modifiez une révision associée à un service, celui-ci est redéployé.
> Vous devez sélectionner Actualiser dans ce panneau pour voir les associations mises à jour.
> Pour connaître l'activité en cours et l'état du redéploiement du service, utilisez le fil de navigation du panneau pour accéder à App Runner > Services, sélectionnez le service, puis consultez l'onglet Logs dans le panneau de présentation des services.



App Runner API or AWS CLI

Utilisez les actions d'API App Runner suivantes pour gérer vos ressources de configuration d'autoscaling.

- CreateAutoScalingConfiguration— Crée une nouvelle configuration de mise à l'échelle automatique ou une révision d'une configuration existante.
- UpdateDefaultAutoScalingConfiguration: définit une configuration de mise à l'échelle automatique comme configuration par défaut. La configuration de mise à l'échelle automatique par défaut existante sera automatiquement définie sur une valeur autre que celle par défaut.
- ListAutoScalingConfigurations— Renvoie la liste des configurations de mise à l'échelle automatique associées à votre Compte AWS, avec des informations récapitulatives.

- ListServicesForAutoScalingConfiguration— Renvoie la liste des services App Runner associés à l'aide d'une configuration de dimensionnement automatique.
- DescribeAutoScalingConfiguration— Renvoie une description complète d'une configuration de mise à l'échelle automatique.
- DeleteAutoScalingConfiguration— Supprime une configuration de dimensionnement automatique. Vous pouvez supprimer une configuration de mise à l'échelle automatique de haut niveau, une révision spécifique d'une configuration ou toutes les révisions associées à la configuration de niveau supérieur. Utilisez le `DeleteAllRevisions` paramètre facultatif pour supprimer toutes les révisions. Si vous atteignez le quota de ressources de configuration de dimensionnement automatique pour votre Compte AWS, vous devrez peut-être supprimer les configurations de mise à l'échelle automatique inutiles.

Gestion des noms de domaine personnalisés pour un service App Runner

Lorsque vous créez un AWS App Runner service, App Runner lui attribue un nom de domaine. Il s'agit d'un sous-domaine du `awsapprunner.com` domaine appartenant à App Runner. Vous pouvez utiliser le nom de domaine pour accéder à l'application Web qui s'exécute dans votre service.

> **Note**
>
> Pour renforcer la sécurité de vos applications App Runner, le domaine*.awsapprunner.com est enregistré dans la liste des suffixes publics (PSL). Pour plus de sécurité, nous vous recommandons d'utiliser des cookies avec un `__Host-` préfixe si vous devez définir des cookies sensibles dans le nom de domaine par défaut de vos applications App Runner. Cette pratique vous aidera à protéger votre domaine contre les tentatives de falsification de requêtes intersites (CSRF). Pour plus d'informations, consultez la page Set-Cookie du Mozilla Developer Network.

Si vous possédez un nom de domaine, vous pouvez l'associer à votre service App Runner. Une fois qu'App Runner a validé votre nouveau domaine, vous pouvez utiliser celui-ci pour accéder à votre application en plus du domaine App Runner. Vous pouvez associer jusqu'à cinq domaines personnalisés.

> **Note**
>
> Vous pouvez éventuellement inclure le `www` sous-domaine de votre domaine. Cependant, cela n'est actuellement pris en charge que par l'API. La console App Runner ne prend pas en charge l'inclusion d'un `www` sous-domaine de votre domaine.

> **Note**
>
> AWS App Runner ne prend pas en charge l'utilisation des zones hébergées privées Route 53. Les zones hébergées privées personnalisent la résolution des noms de domaine pour le trafic Amazon VPC. Pour plus d'informations sur les zones hébergées privées, consultez la section Travailler avec des zones hébergées privées dans la documentation Route 53.

Associer (lien) un domaine personnalisé à votre service

Lorsque vous associez un domaine personnalisé à votre service, vous devez ajouter les enregistrements CNAME et les enregistrements cibles DNS à votre serveur DNS. Les sections suivantes fournissent des informations sur les enregistrements CNAME et les enregistrements cibles DNS, ainsi que sur leur utilisation.

> **Note**
>
> Si vous utilisez Amazon Route 53 comme fournisseur DNS, App Runner configure automatiquement votre domaine personnalisé avec la validation du certificat et les enregistrements DNS requis pour créer un lien vers votre application Web App Runner. Cela se produit lorsque vous utilisez la console App Runner pour associer votre domaine personnalisé à votre service. La Gérez des domaines personnalisés rubrique suivante fournit de plus amples informations.

Enregistrements CNAME

Lorsque vous associez un domaine personnalisé à votre service, App Runner vous fournit un ensemble d'enregistrements de validation de certificat pour la validation des certificats. Vous devez ajouter ces enregistrements de validation de certificat à votre serveur DNS (Domain Name System).

Ajoutez les enregistrements de validation des certificats, fournis par App Runner, à votre serveur DNS. De cette façon, App Runner peut vérifier que vous possédez ou contrôlez le domaine.

> **Note**
>
> Pour renouveler automatiquement vos certificats de domaine personnalisés, assurez-vous de ne pas supprimer les enregistrements de validation des certificats de votre serveur DNS. Pour plus d'informations sur la manière de résoudre les problèmes liés au renouvellement du certificat, consultezthe section called “Renouvellement du certificat de domaine personnalisé”.

App Runner utilise ACM pour vérifier le domaine. Si vous utilisez des enregistrements CAA dans vos enregistrements DNS, assurez-vous qu'au moins un enregistrement CAA fait référence`amazon.com`. Dans le cas contraire, ACM ne pourra pas vérifier le domaine et créer votre domaine avec succès.

Si vous recevez des erreurs liées au CAA, consultez les liens suivants pour savoir comment les résoudre :

- Problèmes d'autorisation de l'autorité de certification (CAA)
- Comment résoudre les erreurs CAA lors de l'émission ou du renouvellement d'un certificat ACM ?
- noms de domaine personnalisés

> **Note**
>
> Si vous utilisez Amazon Route 53 comme fournisseur DNS, App Runner configure automatiquement votre domaine personnalisé avec la validation du certificat et les enregistrements DNS requis pour créer un lien vers votre application Web App Runner. Cela se produit lorsque vous utilisez la console App Runner pour associer votre domaine personnalisé à votre service. La Gérez des domaines personnalisés rubrique suivante fournit de plus amples informations.

Enregistrements cibles DNS

Ajoutez les enregistrements DNS cibles à votre serveur DNS pour cibler le domaine App Runner. Ajoutez un enregistrement pour le domaine personnalisé et un autre pour le `www` sous-domaine, si vous avez choisi cette option. Attendez ensuite que le statut du domaine personnalisé devienne actif

dans la console App Runner. Cela prend généralement plusieurs minutes, mais peut prendre jusqu'à 24 à 48 heures (1 à 2 jours). Lorsque votre domaine personnalisé est validé, App Runner commence à acheminer le trafic de ce domaine vers votre application Web.

> **Note**
>
> Pour une meilleure compatibilité avec les services App Runner, nous vous recommandons d'utiliser Amazon Route 53 comme fournisseur DNS. Si vous n'utilisez pas Amazon Route 53 pour gérer vos enregistrements DNS publics, contactez votre fournisseur DNS pour savoir comment ajouter des enregistrements.
> Si vous utilisez Amazon Route 53 comme fournisseur DNS, vous pouvez ajouter un enregistrement CNAME ou un enregistrement d'alias pour le sous-domaine. Pour le domaine racine, assurez-vous d'utiliser l'enregistrement d'alias.

Vous pouvez acheter un nom de domaine auprès d'Amazon Route 53 ou d'un autre fournisseur. Pour acheter un nom de domaine avec Amazon Route 53, consultez la section Enregistrer un nouveau domaine dans le guide du développeur Amazon Route 53.

Pour savoir comment configurer une cible DNS dans Route 53, consultez la section Router le trafic vers vos ressources dans le manuel Amazon Route 53 Developer Guide.

Pour savoir comment configurer une cible DNS sur d'autres bureaux d'enregistrement, tels que Shopify GoDaddy, Hover, etc., consultez leur documentation spécifique sur l'ajout d'enregistrements DNS Target.

Spécifiez un domaine à associer à votre service App Runner

Vous pouvez spécifier un domaine à associer à votre service App Runner de différentes manières :

- Un domaine racine — Le DNS présente certaines limites inhérentes qui peuvent vous empêcher de créer des enregistrements CNAME pour le nom de domaine racine. Par exemple, si votre nom de domaine est`example.com`, vous pouvez créer un enregistrement CNAME qui achemine le trafic `acme.example.com` vers votre service App Runner. Toutefois, vous ne pouvez pas créer un enregistrement CNAME qui achemine le trafic `example.com` vers votre service App Runner. Pour créer un domaine racine, assurez-vous d'ajouter un enregistrement d'alias.

  Un enregistrement d'alias est spécifique à Route 53 et présente les avantages suivants par rapport aux enregistrements CNAME :

  - Route 53 vous offre plus de flexibilité car des enregistrements d'alias peuvent être créés pour le domaine racine ou le sous-domaine. Par exemple, si votre nom de domaine est`example.com`, vous pouvez créer un enregistrement qui achemine les demandes pour `example.com` ou `acme.example.com` vers votre service App Runner.
  - C'est plus rentable. Cela est dû au fait que Route 53 ne facture pas les demandes qui utilisent un enregistrement d'alias pour acheminer le trafic.
- Un sous-domaine — Par exemple, `login.example.com` ou`admin.login.example.com`. Vous pouvez éventuellement associer le `www` sous-domaine dans le cadre de la même opération. Vous pouvez ajouter un enregistrement CNAME ou un alias pour le sous-domaine.
- Un joker — Par exemple,`*.example.com`. Vous ne pouvez pas utiliser cette `www` option dans ce cas. Vous pouvez spécifier un caractère générique uniquement en tant que sous-domaine immédiat d'un domaine racine et uniquement en tant que tel. Ces spécifications ne sont pas valides :`login*.example.com`,`*.login.example.com`. Cette spécification générique associe tous les sous-domaines immédiats et n'associe pas le domaine racine lui-même. Le domaine racine doit être associé dans le cadre d'une opération distincte.

Une association de domaines plus spécifique remplace une association moins spécifique. Par exemple, les `login.example.com` dérogations`*.example.com`. Le certificat et le CNAME de l'association la plus spécifique sont utilisés.

L'exemple suivant montre comment utiliser plusieurs associations de domaines personnalisées :

1. `example.com`Associez-le à la page d'accueil de votre service. Activez le `www` pour associer`www.example.com`.
2. `login.example.com`Associez-le à la page de connexion de votre service.
3. `*.example.com`Associer à une page personnalisée « introuvable ».

Dissocier (dissocier) un domaine personnalisé

Vous pouvez dissocier (dissocier) un domaine personnalisé de votre service App Runner. Lorsque vous dissociez un domaine, App Runner arrête d'acheminer le trafic de ce domaine vers votre application Web.

> **Note**
>
> Vous devez supprimer les enregistrements du domaine que vous avez dissocié de votre serveur DNS.

App Runner crée en interne des certificats qui suivent la validité du domaine. Ces certificats sont stockés dans AWS Certificate Manager (ACM). App Runner ne supprime pas ces certificats pendant 7 jours après la dissociation d'un domaine de votre service ou après la suppression du service.

Gérez des domaines personnalisés

Gérez les domaines personnalisés pour votre service App Runner à l'aide de l'une des méthodes suivantes :

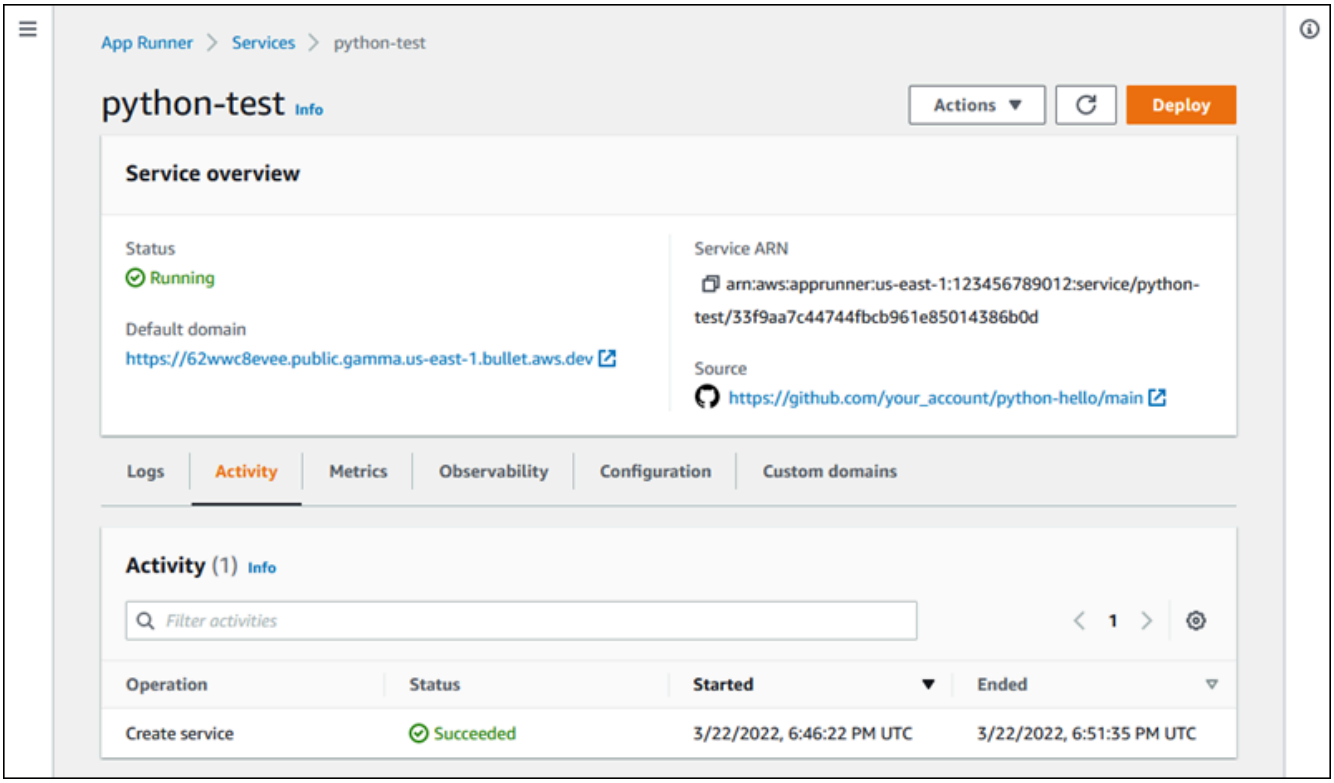
> **Note**
>
> Pour une meilleure compatibilité avec les services App Runner, nous vous recommandons d'utiliser Amazon Route 53 comme fournisseur DNS. Si vous n'utilisez pas Amazon Route 53 pour gérer vos enregistrements DNS publics, contactez votre fournisseur DNS pour savoir comment ajouter des enregistrements.
> Si vous utilisez Amazon Route 53 comme fournisseur DNS, vous pouvez ajouter un enregistrement CNAME ou un enregistrement d'alias pour le sous-domaine. Pour le domaine racine, assurez-vous d'utiliser un enregistrement d'alias.

App Runner console

Pour associer (lier) un domaine personnalisé à l'aide de la console App Runner

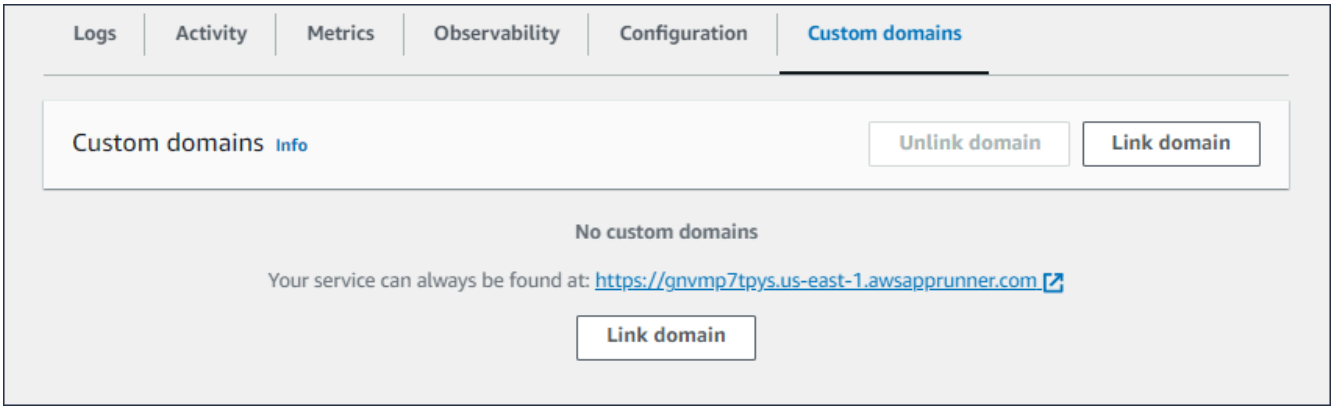
1. Ouvrez la console App Runner, puis dans la liste des régions, sélectionnez votre Région AWS.
2. Dans le volet de navigation, choisissez Services, puis choisissez votre service App Runner.

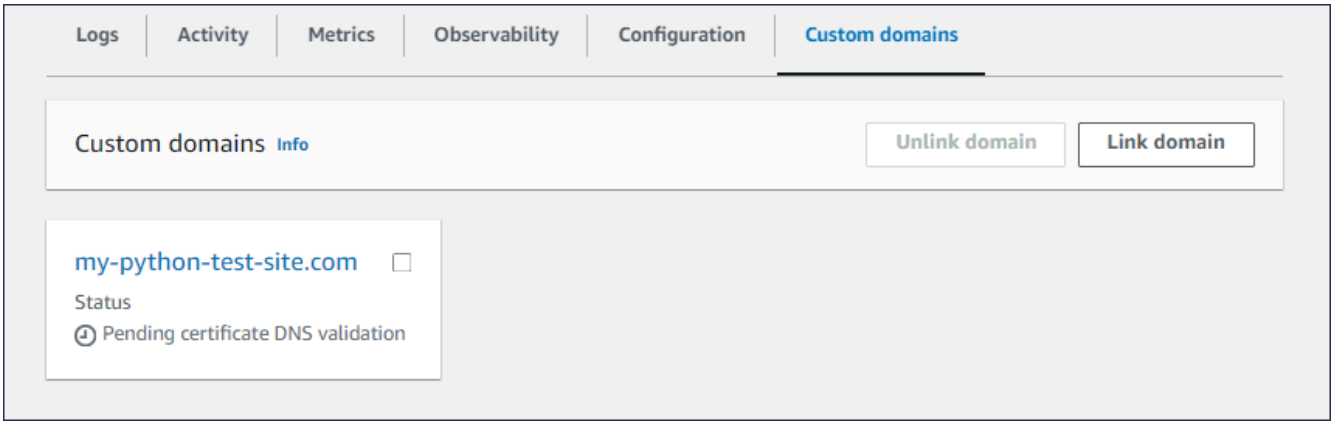
   La console affiche le tableau de bord des services avec une vue d'ensemble des services.



3. Sur la page du tableau de bord du service, choisissez l'onglet Domaines personnalisés.

   La console affiche les domaines personnalisés associés à votre service, ou Aucun domaine personnalisé.

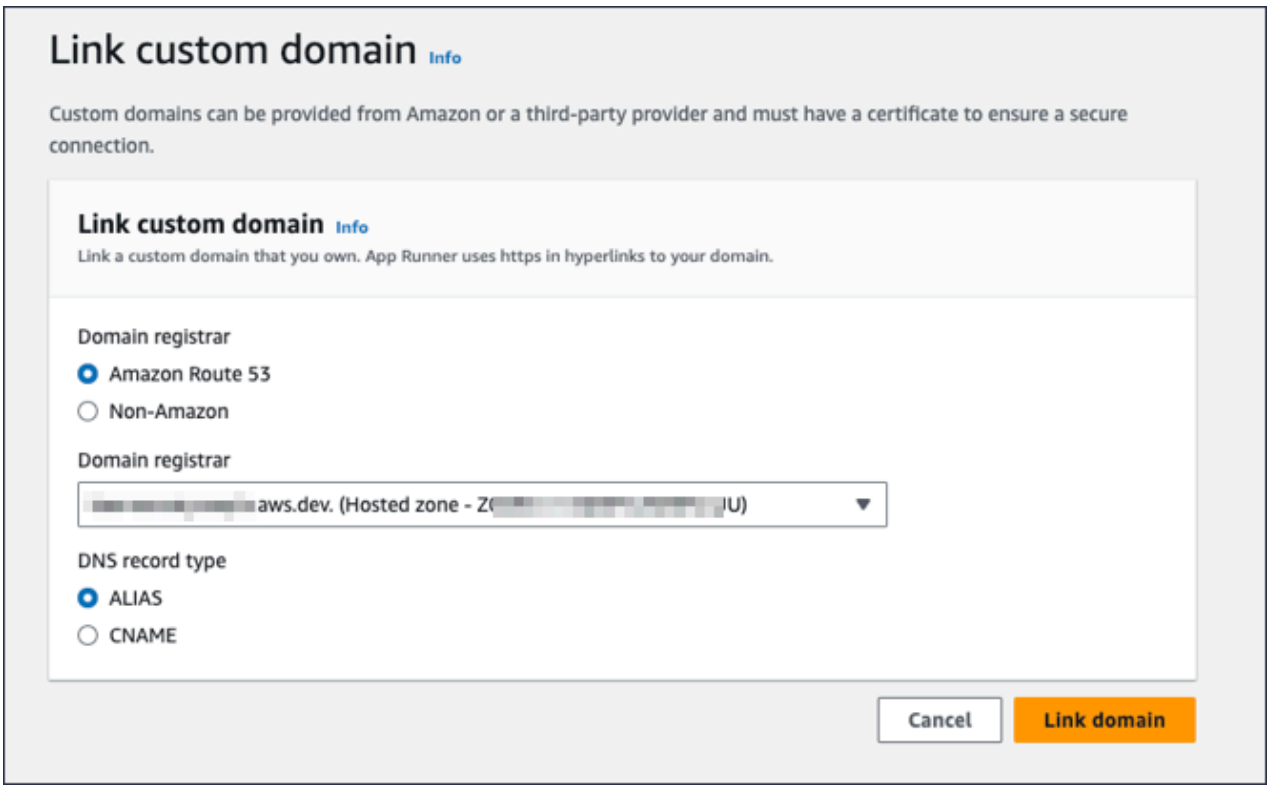




4. Dans l'onglet Domaines personnalisés, choisissez Lier le domaine.
5. La page Lien vers un domaine personnalisé s'affiche.
   - Si votre domaine personnalisé est enregistré auprès d'Amazon Route 53, sélectionnez Amazon Route 53 comme bureau d'enregistrement de domaines.
     a. Sélectionnez le nom de domaine dans la liste déroulante. Cette liste affiche le nom de vos domaines Route 53 et l'identifiant de la zone hébergée.

     > **Note**
     >
     > Vous devez d'abord créer un domaine Route 53 à l'aide du service Amazon Route 53 à partir du même AWS compte que celui que vous utilisez pour gérer vos autres ressources App Runner.

     b. Sélectionnez le type d'enregistrement DNS.
     c. Choisissez le domaine Link.

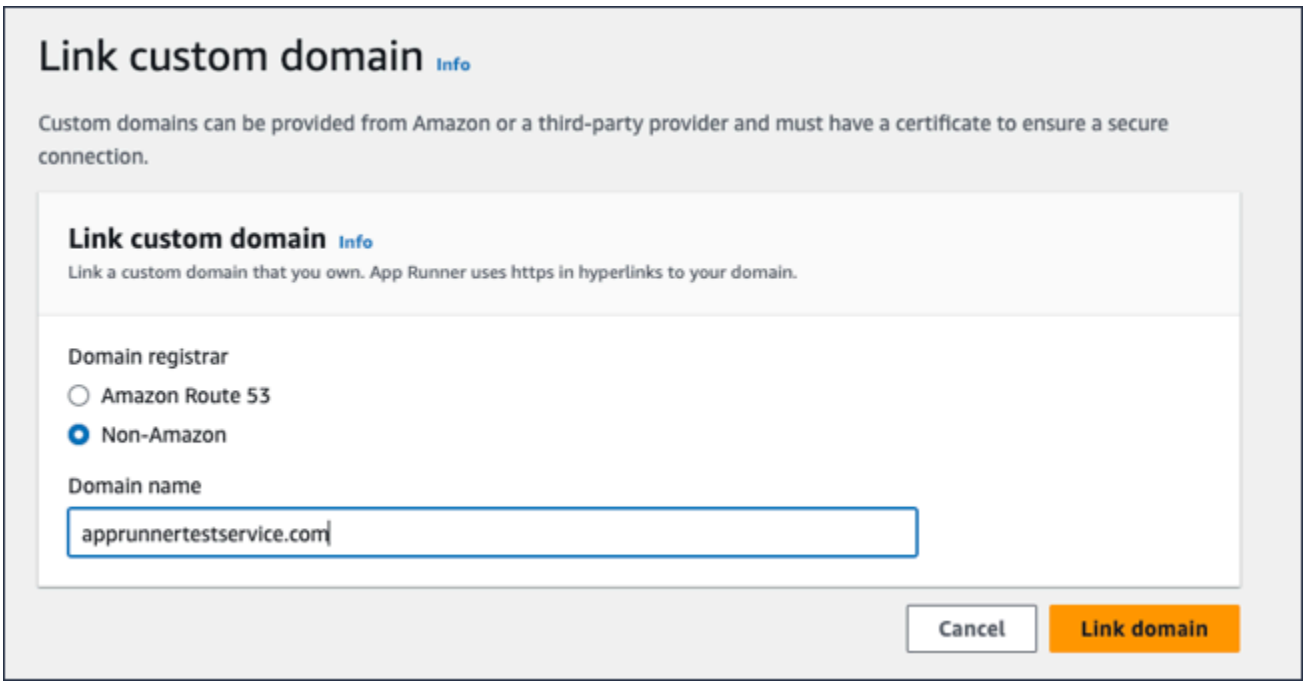


> **Note**
>
> Si App Runner affiche un message d'erreur indiquant que la tentative de configuration automatique a échoué, vous pouvez continuer en configurant les enregistrements DNS manuellement. Ce problème peut survenir si le même nom de domaine a déjà été dissocié d'un service, sans que les enregistrements du fournisseur DNS indiquant que le service n'aient été supprimés par la suite. Dans ce cas, App Runner ne peut pas remplacer automatiquement ces enregistrements. Pour terminer la configuration DNS, sautez les étapes restantes de cette procédure, puis suivez les instructions indiquées dansConfiguration d'un enregistrement d'alias Amazon Route 53.

- Si votre domaine personnalisé est enregistré auprès d'un autre bureau d'enregistrement de domaines, sélectionnez Non-Amazon comme bureau d'enregistrement de domaines.

  a. Entrez le nom de domaine.

  b. Choisissez le domaine Link.



6. La page Configurer le DNS s'affiche.

   - S'il s' Amazon Route 53 agit de votre fournisseur DNS, cette étape est facultative.

     À ce stade, App Runner a automatiquement configuré votre domaine Route 53 avec la validation de certificat et les enregistrements DNS requis.

     > **Note**
     >
     > Si ce même nom de domaine a déjà été dissocié d'un service, sans que le fournisseur DNS enregistre des enregistrements indiquant que le service a été supprimé par la suite, la configuration automatique tentée par App Runner aurait pu échouer. Pour contourner ce problème et terminer l'association DNS, suivez les étapes (1) et (2) de la page Configurer le DNS pour copier les enregistrements de cible et de certificat actuels vers le fournisseur DNS.

   - Copiez les enregistrements de validation des certificats et les enregistrements cibles DNS, puis ajoutez-les à votre serveur DNS. App Runner peut ensuite valider que vous possédez ou contrôlez le domaine.

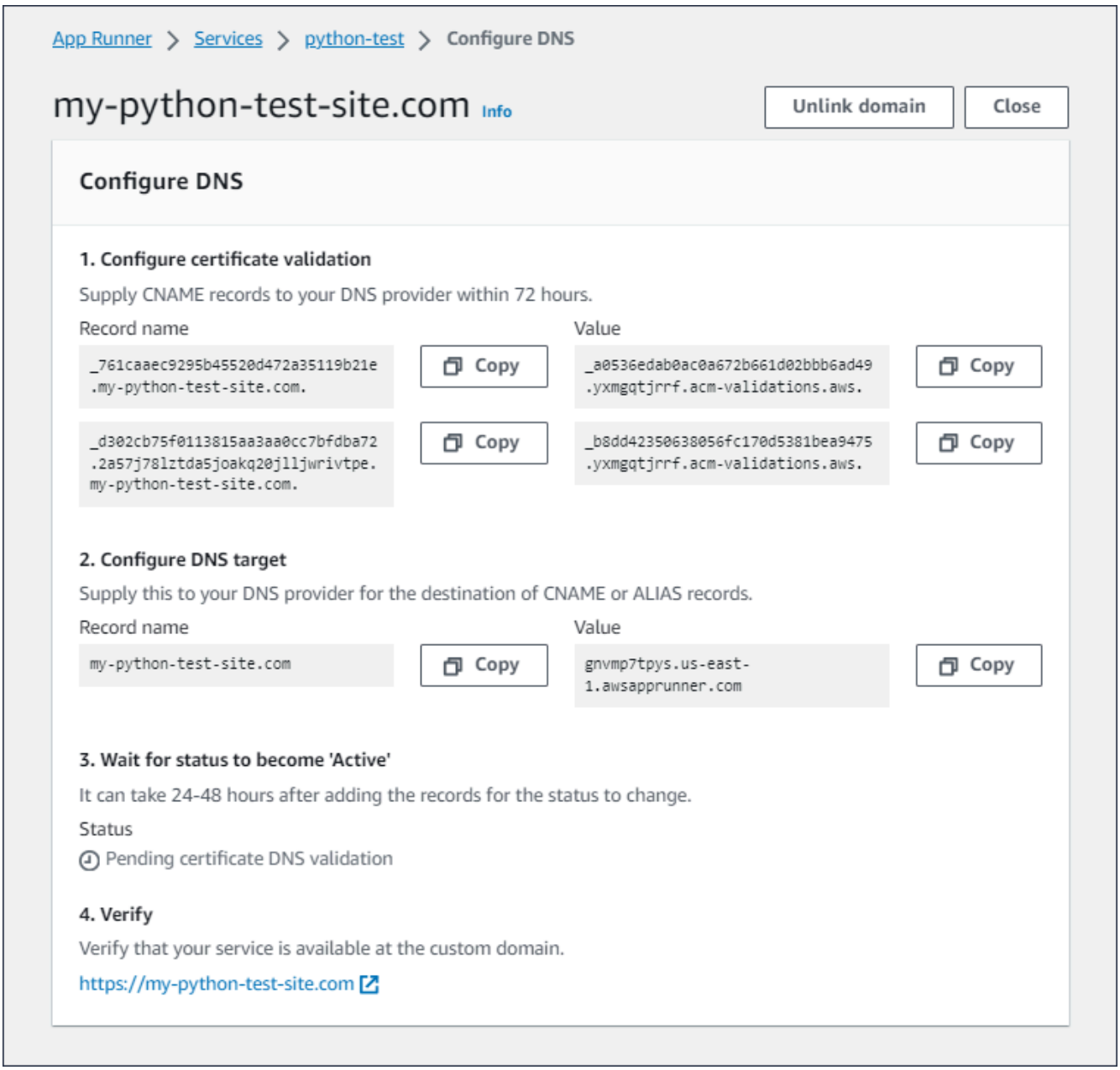
> **Note**
>
> Pour renouveler automatiquement vos certificats de domaine personnalisés, veillez à ne pas supprimer les enregistrements de validation des certificats de votre serveur DNS.

- Pour plus d'informations sur la configuration de la validation des certificats, consultez la section Validation DNS dans le guide de AWS Certificate Manager l'utilisateur.
- Pour plus d'informations sur la configuration d'une cible DNS avec un enregistrement d'alias Amazon Route 53, consultezthe section called “Configuration d'un enregistrement d'alias Amazon Route 53”.

- Si vous utilisez un fournisseur DNS autre qu'Amazon Route 53, procédez comme suit.
  - Copiez les enregistrements de validation des certificats et les enregistrements cibles DNS, puis ajoutez-les à votre serveur DNS. App Runner peut ensuite valider que vous possédez ou contrôlez le domaine.

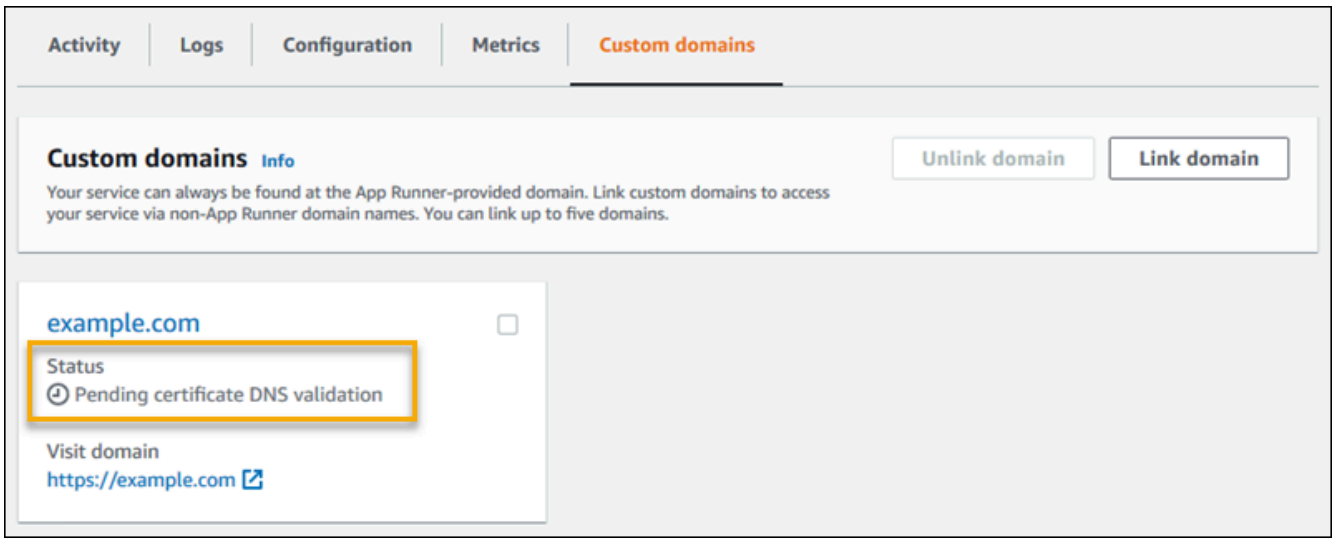
> **Note**
>
> Pour renouveler automatiquement vos certificats de domaine personnalisés, veillez à ne pas supprimer les enregistrements de validation des certificats de votre serveur DNS.

- Pour plus d'informations sur la configuration de la validation des certificats, consultez la section Validation DNS dans le guide de AWS Certificate Manager l'utilisateur.
- Pour savoir comment configurer une cible DNS sur d'autres bureaux d'enregistrement, tels que Shopify GoDaddy, Hover, etc., consultez leur documentation spécifique sur l'ajout d'une cible DNS.

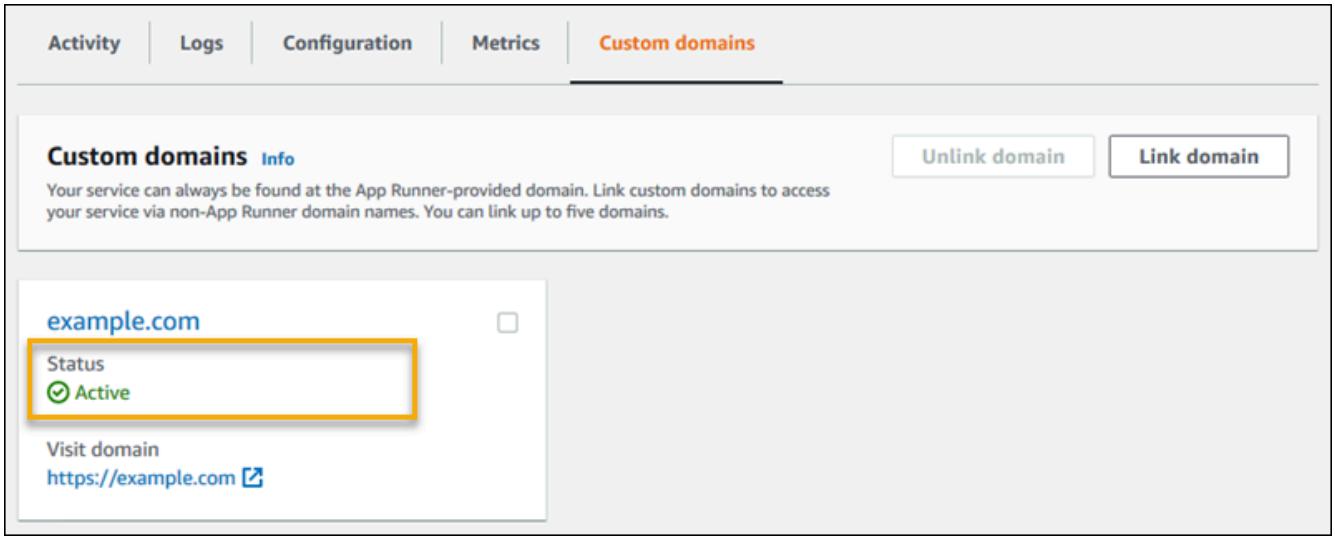


7. Choisissez Fermer

La console affiche à nouveau le tableau de bord. L'onglet Domaines personnalisés comporte une nouvelle vignette indiquant le domaine que vous venez de lier dans l'état de validation DNS du certificat en attente.



8. Lorsque le statut du domaine passe à Actif, vérifiez que le domaine fonctionne pour le routage du trafic en y accédant.



> **Note**
>
> Pour obtenir des instructions sur la résolution des erreurs liées à un domaine personnalisé, consultezthe section called “noms de domaine personnalisés”.

Pour dissocier (dissocier) un domaine personnalisé à l'aide de la console App Runner

1. Dans l'onglet Domaines personnalisés, sélectionnez la vignette du domaine que vous souhaitez dissocier, puis choisissez Dissocier le domaine.

2. Dans la boîte de dialogue Dissocier le domaine, vérifiez l'action en choisissant Dissocier le domaine.

> **Note**
>
> Vous devez supprimer les enregistrements du domaine que vous avez dissocié de votre serveur DNS.

App Runner API or AWS CLI

Pour associer un domaine personnalisé à votre service à l'aide de l'API App Runner ou AWS CLI appelez l'action AssociateCustomDomainAPI. Lorsque l'appel aboutit, un CustomDomainobjet est renvoyé qui décrit le domaine personnalisé associé à votre service. L'objet affiche un `CREATING` statut et contient une liste d'CertificateValidationRecordobjets. L'appel renvoie également l'alias cible que vous pouvez utiliser pour configurer la cible DNS. Il s'agit d'enregistrements que vous pouvez ajouter à votre DNS.

Pour dissocier un domaine personnalisé de votre service à l'aide de l'API App Runner ou AWS CLI appelez l'action DisassociateCustomDomainAPI. Lorsque l'appel aboutit, un CustomDomainobjet est renvoyé qui décrit le domaine personnalisé qui est dissocié de votre service. L'objet affiche un `DELETING` statut.

Rubriques

- Configurer l'enregistrement d'alias Amazon Route 53 pour votre DNS cible

Configurer l'enregistrement d'alias Amazon Route 53 pour votre DNS cible

> **Note**
>
> Il n'est pas nécessaire de suivre cette procédure si Amazon Route 53 est votre fournisseur DNS. Dans ce cas, App Runner configure automatiquement votre domaine Route 53 avec la validation du certificat et les enregistrements DNS requis pour créer un lien vers votre application Web App Runner.
> Si la tentative de configuration automatique d'App Runner échoue, suivez cette procédure pour terminer la configuration DNS. Si le même nom de domaine a déjà été dissocié d'un service, sans que les enregistrements du fournisseur DNS indiquant que le service ne

soient supprimés par la suite, App Runner ne pourra pas remplacer automatiquement ces enregistrements. Cette procédure explique comment les copier manuellement dans votre DNS Route 53.

Vous pouvez utiliser Amazon Route 53 comme fournisseur DNS pour acheminer le trafic vers votre service App Runner. Il s'agit d'un service Web de système de noms de domaine (DNS) hautement disponible et évolutif. L'enregistrement Amazon Route 53 contient les paramètres qui contrôlent la manière dont le trafic est acheminé vers votre service App Runner. Vous créez un enregistrement CNAME ou un enregistrement ALIAS. Pour une comparaison entre les enregistrements CNAME et les enregistrements d'alias, consultez Choisir entre des enregistrements alias et des enregistrements non alias, dans le guide du développeur Amazon Route 53.

Note

Amazon Route 53 prend actuellement en charge l'enregistrement d'alias pour les services créés après le 1er août 2022.

Amazon Route 53 console

Pour configurer l'enregistrement d'alias Amazon Route 53

1. Connectez-vous à la console Route 53 AWS Management Console et ouvrez-la.
2. Dans le panneau de navigation, choisissez Zones hébergées.
3. Choisissez le nom de la zone hébergée que vous souhaitez utiliser pour acheminer le trafic vers votre service App Runner.
4. Choisissez Créer un registre.
5. Indiquez l'une des valeurs suivantes :

   - Politique de routage : Choisissez la politique de routage applicable. Pour plus d'informations, consultez la section Choix d'une politique de routage.
   - Nom de l'enregistrement : entrez le nom de domaine que vous souhaitez utiliser pour acheminer le trafic vers votre service App Runner. La valeur par défaut est le nom de la zone hébergée. Par exemple, si le nom de la zone hébergée est `example.com` et que vous souhaitez l'utiliser `acme.example.com` pour acheminer le trafic vers votre environnement, entrez`acme`.

- Valoriser ou acheminer le trafic vers : choisissez Alias to App Runner Application, puis choisissez la région d'où provient le point de terminaison. Choisissez le nom de domaine de l'application vers laquelle vous souhaitez acheminer le trafic.
- Type d'enregistrement : Acceptez la valeur par défaut, A — IPv4 adresse.
- Évaluer l'état de santé cible : acceptez la valeur par défaut, Oui.

6. Choisissez Créer des enregistrements.

L'enregistrement d'alias Route 53 que vous avez créé est propagé sur tous les serveurs Route 53 en 60 secondes. Lorsque les serveurs Route 53 sont propagés avec votre enregistrement d'alias, vous pouvez acheminer le trafic vers votre service App Runner en utilisant le nom de l'enregistrement d'alias que vous avez créé.

Pour plus d'informations sur la manière de résoudre les problèmes de propagation des modifications DNS, consultez Pourquoi mes modifications de DNS mettent-elles si longtemps à se propager dans Route 53 et dans les résolveurs publics ? .

Amazon Route 53 API or AWS CLI

Pour configurer l'enregistrement d'alias Amazon Route 53 à l'aide de l'API Amazon Route 53 ou pour AWS CLI appeler l'action de l'ChangeResourceRecordSetsAPI. Pour en savoir plus sur l'identifiant de zone hébergée cible de Route 53, consultez la section Points de terminaison du service.

Suspension et reprise d'un service App Runner

Si vous devez désactiver temporairement votre application Web et arrêter l'exécution du code, vous pouvez suspendre votre AWS App Runner service. App Runner réduit la capacité de calcul du service à zéro.

Lorsque vous êtes prêt à exécuter à nouveau votre application, vous pouvez reprendre votre service App Runner. App Runner fournit une nouvelle capacité de calcul, y déploie votre application et l'exécute. La source de votre application n'est pas redéployée et aucune compilation n'est nécessaire. App Runner reprend plutôt avec votre version actuellement déployée. Votre application conserve son domaine App Runner.

Important

- Lorsque vous suspendez votre service, votre application perd son état. Par exemple, tout stockage éphémère utilisé par votre code est perdu. Pour votre code, la suspension et la reprise de votre service équivalent à un déploiement vers un nouveau service.
- Si vous suspendez un service en raison d'une faille dans votre code (par exemple, un bogue découvert ou un problème de sécurité), vous ne pouvez pas déployer une nouvelle version avant de reprendre le service.

  Par conséquent, nous vous recommandons de maintenir le service en cours d'exécution et de revenir à la dernière version stable de votre application.
- Lorsque vous reprenez votre service, App Runner déploie la dernière version de l'application utilisée avant que vous n'interrompiez le service. Si vous avez ajouté de nouvelles versions source depuis la suspension de votre service, App Runner ne les déploie pas automatiquement, même si le déploiement automatique est sélectionné. Supposons, par exemple, que vous ayez de nouvelles versions d'image dans le référentiel d'images ou de nouveaux commits dans le référentiel de code. Ces versions ne sont pas déployées automatiquement.

  Pour déployer une version plus récente, effectuez un déploiement manuel ou ajoutez une autre version à votre référentiel source après avoir repris votre service App Runner.

Comparaison entre pause et suppression

Suspendez votre service App Runner pour le désactiver temporairement. Seules les ressources informatiques sont mises hors service et vos données stockées (par exemple, l'image du conteneur avec la version de votre application) restent intactes. La reprise de votre service est rapide : votre application est prête à être déployée sur de nouvelles ressources informatiques. Votre domaine App Runner reste le même.

Supprimez votre service App Runner pour le supprimer définitivement. Vos données enregistrées sont supprimées. Si vous devez recréer le service, App Runner doit récupérer à nouveau votre source et le créer s'il s'agit d'un dépôt de code. Votre application web obtient un nouveau domaine App Runner.

Lorsque votre service est suspendu

Lorsque vous suspendez votre service et qu'il est dans le statut Suspendu, il répond différemment aux demandes d'action, notamment aux appels d'API ou aux opérations de console. Lorsqu'un service est suspendu, vous pouvez toujours effectuer des actions App Runner qui ne modifient pas la définition ou la configuration du service d'une manière qui affecte son exécution. En d'autres termes, si une action modifie le comportement, l'échelle ou d'autres caractéristiques d'un service en cours d'exécution, vous ne pouvez pas effectuer cette action sur un service suspendu.

Les listes suivantes fournissent des informations sur les actions d'API que vous pouvez et ne pouvez pas effectuer sur un service suspendu. Les opérations de console équivalentes sont autorisées ou refusées de la même manière.

Actions que vous pouvez effectuer sur un service suspendu

- *`List*`*et *`Describe*`* actions : actions qui ne lisent que des informations.
- *`DeleteService`*— Vous pouvez toujours supprimer un service.
- *`TagResource`*, *`UntagResource`* — Les balises sont associées à un service, mais ne font pas partie de sa définition et n'affectent pas son comportement d'exécution.

Actions que vous ne pouvez pas effectuer sur un service suspendu

- *`StartDeployment`*actions (ou déploiement manuel à l'aide de la console)
- *`UpdateService`*(ou une modification de configuration à l'aide de la console, à l'exception des modifications de balisage)
- *`CreateCustomDomainAssociations`*, *`DeleteCustomDomainAssociations`*
- *`CreateConnection`*, *`DeleteConnection`*

Suspendez et reprenez votre service

Suspendez et reprenez votre service App Runner en utilisant l'une des méthodes suivantes :

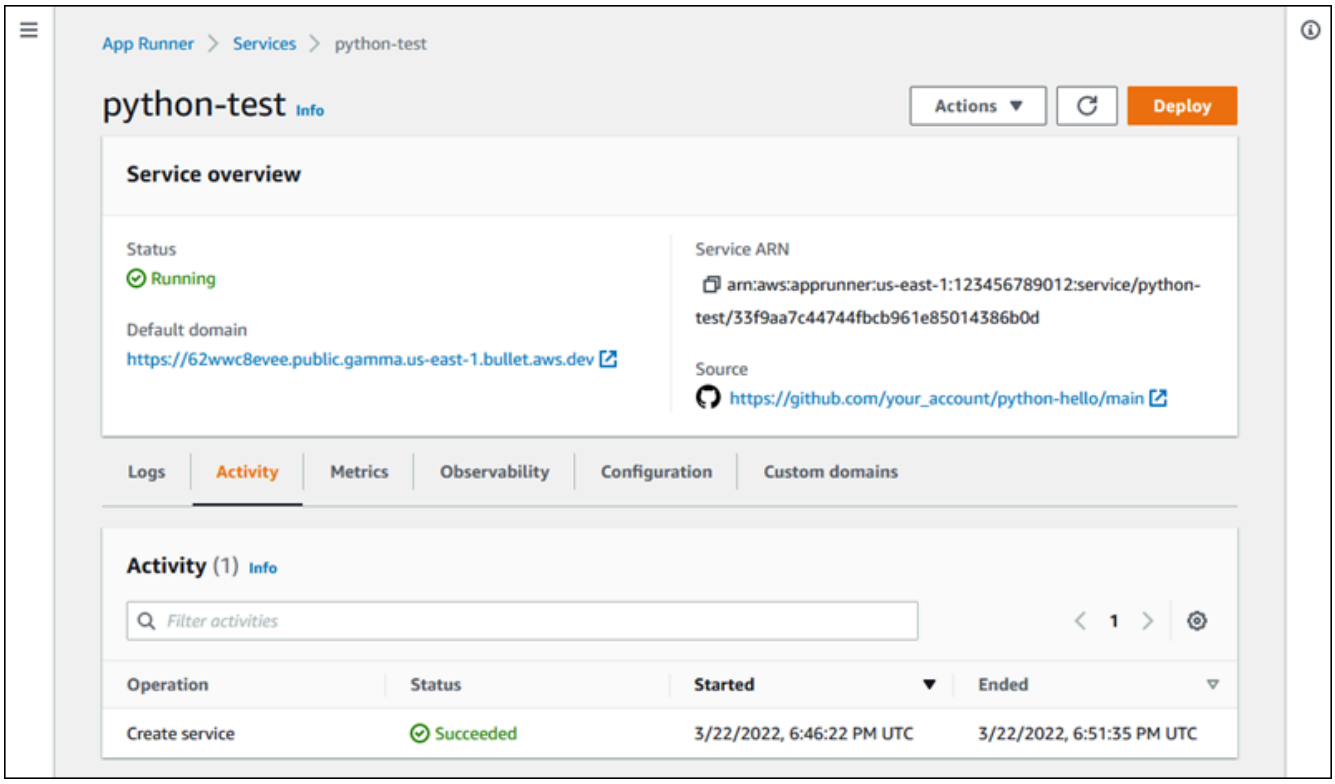
App Runner console

Pour suspendre votre service à l'aide de la console App Runner

1. Ouvrez la console App Runner, puis dans la liste des régions, sélectionnez votre Région AWS.

2. Dans le volet de navigation, choisissez Services, puis choisissez votre service App Runner.

   La console affiche le tableau de bord des services avec une vue d'ensemble des services.



3. Choisissez Actions, puis sélectionnez Pause.

   Sur la page du tableau de bord du service, le statut du service passe à Opération en cours, puis à Suspendu. Votre service est maintenant suspendu.

Pour reprendre votre service à l'aide de la console App Runner

1. Choisissez Actions, puis Resume.

   Sur la page du tableau de bord du service, l'état du service devient Opération en cours.

2. Attendez que le service reprenne. Sur la page du tableau de bord du service, le statut du service redevient En cours d'exécution.

3. Pour vérifier que la reprise du service est réussie, sur la page du tableau de bord du service, choisissez la valeur du domaine App Runner. Il s'agit de l'URL du site Web de votre service. Vérifiez que votre application Web fonctionne correctement.

App Runner API or AWS CLI

Pour suspendre votre service à l'aide de l'API App Runner ou AWS CLI appelez l'action PauseServiceAPI. Si l'appel renvoie une réponse positive avec un objet Service affiché`"Status": "OPERATION_IN_PROGRESS"`, App Runner commence à suspendre votre service.

Pour reprendre votre service à l'aide de l'API App Runner ou AWS CLI appelez l'action ResumeServiceAPI. Si l'appel renvoie une réponse positive avec un objet Service affiché`"Status": "OPERATION_IN_PROGRESS"`, App Runner commence à reprendre votre service.

Supprimer un service App Runner

Lorsque vous souhaitez arrêter l'application Web en cours d'exécution dans votre AWS App Runner service, vous pouvez supprimer le service. La suppression d'un service arrête le service Web en cours d'exécution, supprime les ressources sous-jacentes et supprime les données associées.

Vous souhaiterez peut-être supprimer un service App Runner pour une ou plusieurs des raisons suivantes :

- Vous n'avez plus besoin de l'application Web. Par exemple, elle a été retirée ou il s'agit d'une version de développement que vous n'utilisez plus.
- Vous avez atteint le quota de service d'App Runner. Vous souhaitez créer un nouveau service dans le même Région AWS environnement et vous avez atteint le quota associé à votre compte. Pour de plus amples informations, veuillez consulter the section called “Quotas de ressources App Runner”.
- Considérations relatives à la sécurité ou à la confidentialité : vous souhaitez qu'App Runner supprime les données stockées pour votre service.

Comparaison entre pause et suppression

Suspendez votre service App Runner pour le désactiver temporairement. Seules les ressources informatiques sont mises hors service et vos données stockées (par exemple, l'image du conteneur avec la version de votre application) restent intactes. La reprise de votre service est rapide : votre application est prête à être déployée sur de nouvelles ressources informatiques. Votre domaine App Runner reste le même.

Supprimez votre service App Runner pour le supprimer définitivement. Vos données enregistrées sont supprimées. Si vous devez recréer le service, App Runner doit récupérer à nouveau votre source et le créer s'il s'agit d'un dépôt de code. Votre application web obtient un nouveau domaine App Runner.

Que supprime App Runner ?

Lorsque vous supprimez votre service, App Runner supprime certains éléments associés, mais n'en supprime pas d'autres. Les listes suivantes fournissent des informations détaillées.

Éléments supprimés par App Runner :

- Image du conteneur : copie de l'image que vous avez déployée ou de l'image créée par App Runner à partir de votre code source. Il est stocké dans Amazon Elastic Container Registry (Amazon ECR) à l'aide de fichiers Comptes AWS internes appartenant à App Runner.
- Configuration du service : paramètres de configuration associés à votre service App Runner. Ils sont stockés dans Amazon DynamoDB à l'aide de fichiers Comptes AWS internes appartenant à App Runner.

Éléments qu'App Runner ne supprime pas :

- Connexion : vous disposez peut-être d'une connexion associée à votre service. Une connexion App Runner est une ressource distincte qui peut être partagée entre plusieurs services App Runner. Si vous n'avez plus besoin de la connexion, vous pouvez la supprimer explicitement. Pour de plus amples informations, veuillez consulter the section called “Connexions”.
- Certificats de domaine personnalisés : si vous liez des domaines personnalisés à un service App Runner, App Runner crée en interne des certificats qui suivent la validité du domaine. Ils sont stockés dans AWS Certificate Manager (ACM). App Runner ne supprime pas le certificat pendant sept jours après la dissociation d'un domaine de votre service ou après la suppression du service. Pour de plus amples informations, veuillez consulter the section called “noms de domaine personnalisés”.

Supprimer votre service

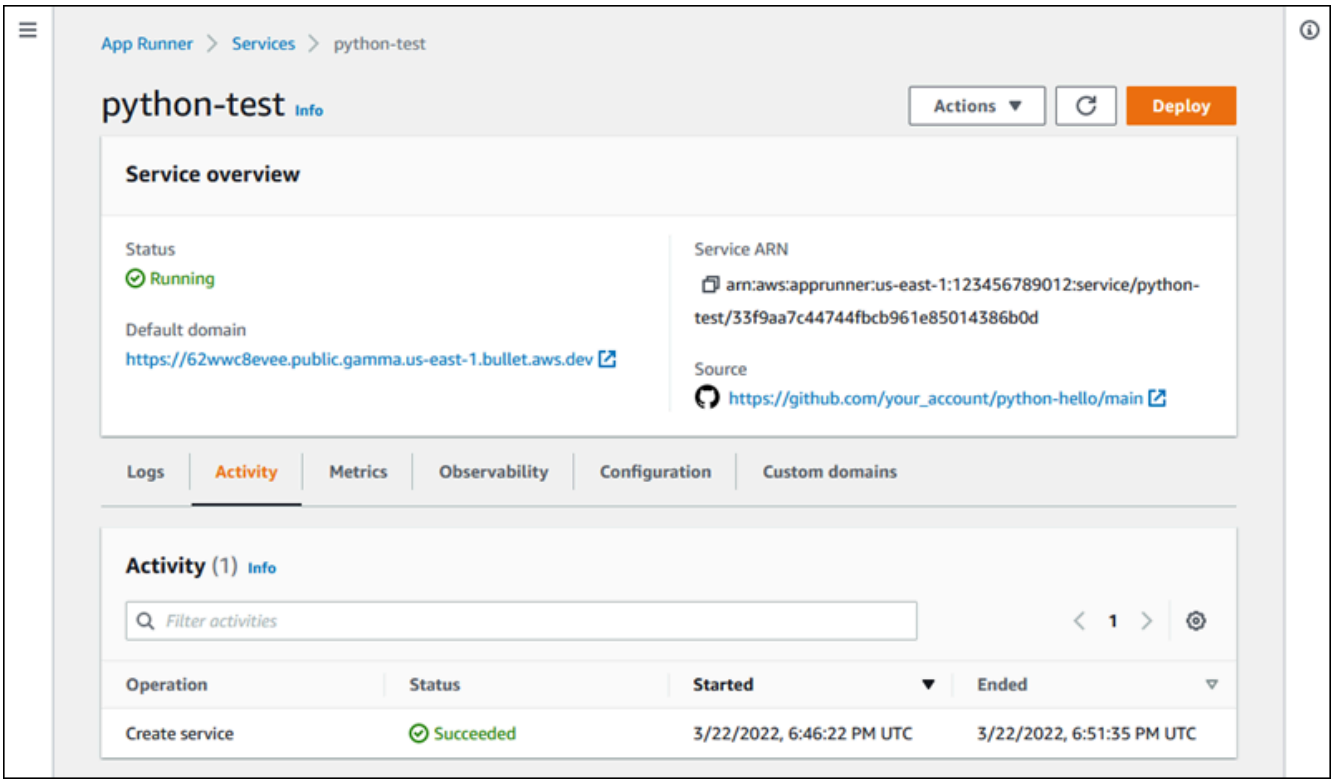
Supprimez votre service App Runner en utilisant l'une des méthodes suivantes :

App Runner console

Pour supprimer votre service à l'aide de la console App Runner

1. Ouvrez la console App Runner, puis dans la liste des régions, sélectionnez votre Région AWS.
2. Dans le volet de navigation, choisissez Services, puis choisissez votre service App Runner.

   La console affiche le tableau de bord des services avec une vue d'ensemble des services.



3. Choisissez Actions, puis Supprimer.

   La console vous amène à la page Services. Le service supprimé affiche le statut Opération en cours, puis il disparaît de la liste. Votre service est maintenant supprimé.

App Runner API or AWS CLI

Pour supprimer votre service à l'aide de l'API App Runner ou AWS CLI appelez l'action DeleteServiceAPI. Si l'appel renvoie une réponse positive avec un objet Service affiché`"Status": "OPERATION_IN_PROGRESS"`, App Runner commence à supprimer votre service.

Référencement des variables d'environnement

Avec App Runner, vous pouvez référencer des secrets et des configurations en tant que variables d'environnement dans votre service lorsque vous créez un service ou que vous le mettez à jour.

Vous pouvez référencer des données de configuration non sensibles telles que les délais d'expiration et le nombre de nouvelles tentatives en texte brut sous forme de paires clé-valeur. Les données de configuration auxquelles vous faites référence en texte brut ne sont pas cryptées et sont visibles par les autres utilisateurs dans les journaux de configuration du service et des applications App Runner.

> **Note**
>
> Pour des raisons de sécurité, ne faites référence à aucune donnée sensible en texte brut dans votre service App Runner.

Référencement de données sensibles en tant que variables d'environnement

App Runner permet de référencer en toute sécurité les données sensibles en tant que variables d'environnement dans votre service. Envisagez de stocker les données sensibles que vous souhaitez référencer dans le AWS Secrets ManagerAWS Systems Manager Parameter Store. Vous pouvez ensuite les référencer en toute sécurité dans votre service en tant que variables d'environnement depuis la console App Runner ou en appelant l'API. Cela permet de séparer efficacement la gestion des secrets et des paramètres du code de votre application et de la configuration des services, améliorant ainsi la sécurité globale de vos applications exécutées sur App Runner.

> **Note**
>
> App Runner ne vous facture pas le fait de référencer Secrets Manager et SSM Parameter Store en tant que variables d'environnement. Toutefois, vous payez le tarif standard pour utiliser Secrets Manager et SSM Parameter Store.
> Pour plus d'informations sur les tarifs, consultez les rubriques suivantes :
>
> - AWS Tarifs de Secrets Manager

- AWS Tarification du SSM Parameter Store

Le processus de référence des données sensibles en tant que variables d'environnement est le suivant :

1. Stockez les données sensibles, telles que les clés d'API, les informations d'identification de base de données, les paramètres de connexion à la base de données ou les versions des applications sous forme de secrets ou de paramètres dans l'un AWS Secrets Manager ou l'autre magasin de AWS Systems Manager paramètres.
2. Mettez à jour la politique IAM de votre rôle d'instance afin qu'App Runner puisse accéder aux secrets et aux paramètres stockés dans Secrets Manager et SSM Parameter Store. Pour plus d'informations, consultez Autorisations .
3. Référencez de manière sécurisée les secrets et les paramètres en tant que variables d'environnement en leur attribuant un nom et en fournissant leur Amazon Resource Name (ARN). Vous pouvez ajouter des variables d'environnement lorsque vous créez un service ou que vous mettez à jour la configuration d'un service. Vous pouvez utiliser l'une des options suivantes pour ajouter des variables d'environnement :
   - Console App Runner
   - API App Runner
   - Fichier de configuration `apprunner.yaml`

   > **Note**
   >
   > Vous ne pouvez pas attribuer `PORT` de nom à une variable d'environnement lors de la création ou de la mise à jour de votre service App Runner. Il s'agit d'une variable d'environnement réservée au service App Runner.

   Pour plus d'informations sur la façon de référencer les secrets et les paramètres, consultez la section Gestion des variables d'environnement.

> **Note**
>
> Comme App Runner ne stocke que la référence au secret et au paramètre ARNs, les données sensibles ne sont pas visibles par les autres utilisateurs dans la configuration du service App Runner et dans les journaux des applications.

Considérations

- Assurez-vous de mettre à jour votre rôle d'instance avec les autorisations appropriées pour accéder aux secrets et aux paramètres dans AWS Secrets Manager ou dans AWS Systems Manager Parameter Store. Pour plus d'informations, consultez Autorisations .
- Assurez-vous que AWS Systems Manager Parameter Store se trouve dans le même Compte AWS que le service que vous souhaitez lancer ou mettre à jour. À l'heure actuelle, vous ne pouvez pas référencer les paramètres du magasin de paramètres SSM entre les comptes.
- Lorsque les secrets et les valeurs des paramètres sont pivotés ou modifiés, ils ne sont pas automatiquement mis à jour dans votre service App Runner. Redéployez votre service App Runner car App Runner extrait uniquement les secrets et les paramètres lors du déploiement.
- Vous avez également la possibilité d'appeler directement un AWS Secrets Manager AWS Systems Manager Parameter Store via le SDK de votre service App Runner.
- Pour éviter les erreurs, veillez à respecter les points suivants lorsque vous les référencez en tant que variables d'environnement :
  - Vous spécifiez le bon ARN du secret.
  - Vous spécifiez le nom ou le bon ARN du paramètre.

Permissions

Pour activer le référencement des secrets et des paramètres stockés dans le magasin de paramètres AWS Secrets Manager ou SSM, ajoutez les autorisations appropriées à la politique IAM de votre rôle d'instance pour accéder à Secrets Manager et au magasin de paramètres SSM.

> **Note**
>
> App Runner ne peut pas accéder aux ressources de votre compte sans votre autorisation. Vous fournissez l'autorisation en mettant à jour votre politique IAM.

Vous pouvez utiliser les modèles de politique suivants pour mettre à jour votre rôle d'instance dans la console IAM. Vous pouvez modifier ces modèles de politique pour répondre à vos besoins spécifiques. Pour plus d'informations sur la mise à jour d'un rôle d'instance, consultez la section Modification d'un rôle dans le Guide de l'utilisateur IAM.

> **Note**
>
> Vous pouvez également copier les modèles suivants depuis la console App Runner lors de la création des variables d'environnement.

Copiez le modèle suivant dans votre rôle d'instance pour ajouter l'autorisation de référencer les secrets à partir de AWS Secrets Manager.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "secretsmanager:GetSecretValue",
            "kms:Decrypt*"
        ],
        "Resource": [
            "arn:aws:secretsmanager:us-east-1:111122223333:secret:my-secret",
            "arn:aws:kms:us-east-1:111122223333:key/my-key"
         ]
     }
   ]
}
```

Copiez le modèle suivant dans votre rôle d'instance pour ajouter l'autorisation de référencer les paramètres depuis AWS Systems ManagerParameter Store.

JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameters"
            ],
            "Resource": [
            "arn:aws:ssm:us-east-1:111122223333:parameter/my-parameter"
            ]
        }
    ]
}
```

Gestion de vos variables d'environnement

Gérez les variables d'environnement de votre service App Runner à l'aide de l'une des méthodes suivantes :

- the section called “Console App Runner”
- the section called “API App Runner ou AWS CLI”

Console App Runner

Lorsque vous créez un service ou que vous mettez à jour un service sur la console App Runner, vous pouvez ajouter des variables d'environnement.

Ajout d'une variable d'environnement

Pour ajouter une variable d'environnement

1. Ouvrez la console App Runner, puis dans la liste des régions, sélectionnez votre Région AWS.
2. Selon que vous créez ou mettez à jour un service, effectuez l'une des étapes suivantes :
    - Si vous créez un nouveau service, choisissez Create an App Runner service et accédez à Configurer le service.
    - Si vous mettez à jour un service existant, sélectionnez le service que vous souhaitez mettre à jour et accédez à l'onglet Configuration du service.

3. Accédez à Variables d'environnement (facultatif) sous Paramètres du service.
4. Choisissez l'une des options suivantes en fonction de vos besoins :

   - Choisissez Texte brut dans la source de la variable d'environnement et entrez ses paires clé-valeur sous Nom de la variable d'environnement et Valeur de la variable d'environnement, respectivement.

     > **Note**
     >
     > Choisissez le texte brut si vous souhaitez référencer des données non sensibles. Ces données ne sont pas cryptées et sont visibles par les autres utilisateurs dans la configuration du service App Runner et dans les journaux des applications.

   - Choisissez Secrets Manager dans la source de la variable d'environnement pour référencer le secret stocké en AWS Secrets Manager tant que variable d'environnement dans votre service. Indiquez le nom de la variable d'environnement et le nom de ressource Amazon (ARN) du secret auquel vous faites référence sous Nom de la variable d'environnement et Valeur de la variable d'environnement respectivement.
   - Choisissez le magasin de paramètres SSM dans la source de la variable d'environnement pour référencer le paramètre stocké dans le magasin de paramètres SSM en tant que variable d'environnement dans votre service. Indiquez le nom de la variable d'environnement et l'ARN du paramètre que vous référencez sous Nom de la variable d'environnement et Valeur de la variable d'environnement respectivement.

     > **Note**
     >
     > - Vous ne pouvez pas attribuer `PORT` de nom à une variable d'environnement lors de la création ou de la mise à jour de votre service App Runner. Il s'agit d'une variable d'environnement réservée au service App Runner.
     > - Si le paramètre SSM Parameter Store est Région AWS identique à celui du service que vous souhaitez lancer, vous pouvez spécifier le nom complet de la ressource Amazon (ARN) ou le nom du paramètre. Si le paramètre se trouve dans une autre région, vous devez spécifier l'ARN complet.
     > - Assurez-vous que le paramètre auquel vous faites référence se trouve dans le même compte que le service que vous lancez ou mettez à jour. À l'heure actuelle,

vous ne pouvez pas référencer le paramètre SSM Parameter Store entre les comptes.

5. Choisissez Ajouter une variable d'environnement pour faire référence à une autre variable d'environnement.
6. Développez les modèles de stratégie IAM pour afficher et copier les modèles de politique IAM fournis pour le magasin de paramètres AWS Secrets Manager et SSM. Vous ne devez le faire que si vous n'avez pas encore mis à jour la politique IAM de votre rôle d'instance avec les autorisations requises. Pour plus d'informations, consultez Autorisations .

Suppression d'une variable d'environnement

Avant de supprimer une variable d'environnement, assurez-vous que le code de votre application est mis à jour pour refléter la même variable. Si le code de l'application n'est pas mis à jour, votre service App Runner risque d'échouer.

Pour supprimer des variables d'environnement

1. Ouvrez la console App Runner, puis dans la liste des régions, sélectionnez votre Région AWS.
2. Accédez à l'onglet Configuration du service que vous souhaitez mettre à jour.
3. Accédez à Variables d'environnement (facultatif) sous Paramètres du service.
4. Choisissez Supprimer à côté de la variable d'environnement que vous souhaitez supprimer. Vous recevez un message pour confirmer la suppression.
5. Sélectionnez Delete (Supprimer).

API App Runner ou AWS CLI

Vous pouvez référencer des données sensibles stockées dans Secrets Manager et SSM Parameter Store en les ajoutant en tant que variables d'environnement dans votre service.

> **Note**
>
> Mettez à jour la politique IAM de votre rôle d'instance afin qu'App Runner puisse accéder aux secrets et aux paramètres stockés dans Secrets Manager et SSM Parameter Store. Pour plus d'informations, consultez Autorisations .

Pour référencer les secrets et les configurations en tant que variables d'environnement

1. Créez un secret ou une configuration dans le Secrets Manager ou le magasin de paramètres SSM.

   Les exemples suivants montrent comment créer un secret et un paramètre à l'aide du magasin de paramètres SSM.

   Example Création d'un secret - Demande

   L'exemple suivant montre comment créer un secret qui représente les informations d'identification de la base de données.

   ```
   aws secretsmanager create-secret \
   –name DevRdsCredentials \
   –description "Rds credentials for development account." \
   –secret-string "{\"user\":\"diegor\",\"password\":\"EXAMPLE-PASSWORD\"}"
   ```

   Example Création d'un secret - Réponse

   ```
   arn:aws:secretsmanager:<region>:<aws_account_id>:secret:DevRdsCredentials
   ```

   Example Création d'une configuration - Demande

   L'exemple suivant montre comment créer un paramètre qui représente la chaîne de connexion RDS.

   ```
   aws systemsmanager put-parameter \
   –name DevRdsConnectionString \
   –value "mysql2://dev-mysqlcluster-rds.com:3306/diegor" \
   –type "String" \
   –description "Rds connection string for development account."
   ```

   Example Création d'une configuration - Réponse

   ```
   arn:aws:ssm:<region>:<aws_account_id>:parameter/DevRdsConnectionString
   ```

2. Référencez les secrets et les configurations stockés dans Secrets Manager et SSM Parameter Store en les ajoutant en tant que variables d'environnement. Vous pouvez ajouter des variables d'environnement lorsque vous créez ou mettez à jour votre service App Runner.

Les exemples suivants montrent comment référencer des secrets et des configurations en tant que variables d'environnement sur un service App Runner basé sur du code et sur un service App Runner basé sur des images.

Example Fichier Input.json pour le service App Runner basé sur des images

```
{
  "ServiceName": "example-secrets",
  "SourceConfiguration": {
    "ImageRepository": {
      "ImageIdentifier": "<image-identifier>",
      "ImageConfiguration": {
        "Port": "<port>",
        "RuntimeEnvironmentSecrets": {

 "Credential1":"arn:aws:secretsmanager:<region>:<aws_account_id>:secret:XXXXXXXXXXXX",
          "Credential2":"arn:aws:ssm:<region>:<aws_account_id>:parameter/
<parameter-name>"
        }
      },
      "ImageRepositoryType": "ECR_PUBLIC"
    }
  },
  "InstanceConfiguration": {
    "Cpu": "1 vCPU",
    "Memory": "3 GB",
    "InstanceRoleArn": "<instance-role-arn>"
  }
}
```

Example Service App Runner basé sur des images — Demande

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

Example Service App Runner basé sur des images — Response

```
{
...
      "ImageRepository": {
         "ImageIdentifier":"<image-identifier>",
```

```
         "ImageConfiguration":{
            "Port": "<port>",
            "RuntimeEnvironmentSecrets":{
               "Credential1":
 "arn:aws:secretsmanager:<region>:<aws_account_id>:secret:XXXXXXXXXXXX",
               "Credential2": "arn:aws:ssm:<region>:<aws_account_id>:parameter/
<parameter-name>"
            },
            "ImageRepositoryType":"ECR"
      }
   },
   "InstanceConfiguration": {
        "CPU": "1 vCPU",
        "Memory": "3 GB",
        "InstanceRoleArn: "<instance-role-arn>"
   }
...
}
```

Example Fichier Input.json pour le service App Runner basé sur du code

```
{
  "ServiceName": "example-secrets",
  "SourceConfiguration": {
    "AuthenticationConfiguration": {
      "ConnectionArn": "arn:aws:apprunner:us-east-1:123456789012:connection/my-
github-connection/XXXXXXXXXX"
    },
    "AutoDeploymentsEnabled": false,
    "CodeRepository": {
      "RepositoryUrl": "<repository-url>",
      "SourceCodeVersion": {
        "Type": "BRANCH",
        "Value": "main"
      },
      "CodeConfiguration": {
        "ConfigurationSource": "API",
        "CodeConfigurationValues": {
          "Runtime": "<runtime>",
          "BuildCommand": "<build-command>",
          "StartCommand": "<start-command>",
          "Port": "<port>",
          "RuntimeEnvironmentSecrets": {
```

```
 "Credential1":"arn:aws:secretsmanager:<region>:<aws_account_id>:secret:XXXXXXXXXXXX",
           "Credential2":"arn:aws:ssm:<region>:<aws_account_id>:parameter/
<parameter-name>"
         }
       }
     }
   }
  },
  "InstanceConfiguration": {
    "Cpu": "1 vCPU",
    "Memory": "3 GB",
    "InstanceRoleArn": "<instance-role-arn>"
  }
}
```

Example Service App Runner basé sur du code — Demande

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

Example Service App Runner basé sur du code — Response

```
{
...
   "SourceConfiguration":{
      "CodeRepository":{
         "RepositoryUrl":"<repository-url>",
         "SourceCodeVersion":{
            "Type":"Branch",
            "Value":"main"
         },
         "CodeConfiguration":{
            "ConfigurationSource":"API",
            "CodeConfigurationValues":{
               "Runtime":"<runtime>",
               "BuildCommand":"<build-command>",
               "StartCommand":"<start-command>",
               "Port":"<port>",
               "RuntimeEnvironmentSecrets":{
                  "Credential1" :
 "arn:aws:secretsmanager:<region>:<aws_account_id>:secret:XXXXXXXX",
```

```
                  "Credential2" : "arn:aws:ssm:<region>:<aws_account_id>:parameter/
<parameter-name>"
               }
            }
         }
      },
      "InstanceConfiguration": {
        "CPU": "1 vCPU",
        "Memory": "3 GB",
        "InstanceRoleArn: "<instance-role-arn>"
   }
...
}
```

3. Le `apprunner.yaml` modèle est mis à jour pour refléter les secrets ajoutés.

   Voici un exemple du `apprunner.yaml` modèle mis à jour.

   Example **`apprunner.yaml`**

```
version: 1.0
runtime: python3
build:
  commands:
    build:
      - python -m pip install flask
run:
  command: python app.py
  network:
    port: 8080
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
  secrets:
    - name: my-secret
      value-from:
 "arn:aws:secretsmanager:<region>:<aws_account_id>:secret:XXXXXXXXXXXX"
    - name: my-parameter
      value-from: "arn:aws:ssm:<region>:<aws_account_id>:parameter/<parameter-
name>"
    - name: my-parameter-only-name
      value-from: "parameter-name"
```

Mise en réseau avec App Runner

Ce chapitre décrit les configurations réseau de vos AWS App Runner services.

Dans ce chapitre, vous apprendrez ce qui suit :

- Comment configurer votre trafic entrant pour les points de terminaison privés et publics. Pour plus d'informations, voir Configuration des configurations réseau pour le trafic entrant.
- Comment configurer votre trafic sortant pour accéder à d'autres applications exécutées dans un Amazon VPC. Pour plus d'informations, consultez Activation de l'accès VPC pour le trafic sortant.

Rubriques

- Terminologie
- Configuration des configurations réseau pour le trafic entrant
- Activation de l'accès VPC pour le trafic sortant

Terminologie

Pour savoir comment personnaliser le trafic de votre réseau en fonction de vos besoins, comprenons les termes suivants utilisés dans ce chapitre.

Conditions générales

Pour savoir ce qu'il faut associer à un Amazon Virtual Private Cloud (VPC), comprenons les termes suivants :

- VPC : un Amazon VPC est un réseau virtuel isolé de manière logique qui vous permet de contrôler totalement votre environnement réseau virtuel, y compris le placement des ressources, la connectivité et la sécurité. Il s'agit d'un réseau virtuel qui ressemble beaucoup à un réseau traditionnel que vous exploiteriez dans votre propre centre de données.
- Point de terminaison de l'interface VPC : le point de terminaison de l'interface VPC, une AWS PrivateLink ressource, connecte un VPC à un service de point de terminaison. Créez un point de terminaison d'interface VPC pour envoyer le trafic aux services de point de terminaison qui utilisent un Network Load Balancer pour distribuer le trafic. Le trafic destiné au service de point de terminaison est résolu à l'aide du DNS.

- Régions : chaque région est une zone géographique distincte dans laquelle vous pouvez héberger un service App Runner.
- Zones de disponibilité : Une zone de disponibilité est un emplacement isolé au sein d'une AWS région. Il s'agit d'un ou de plusieurs centres de données distincts dotés d'une alimentation, d'un réseau et d'une connectivité redondants. Les zones de disponibilité vous aident à rendre les applications de production hautement disponibles, tolérantes aux pannes et évolutives.
- Sous-réseaux : un sous-réseau est une plage d'adresses IP de votre VPC. Un sous-réseau doit se trouver dans une seule zone de disponibilité. Vous pouvez lancer une AWS ressource dans un sous-réseau spécifique. Utilisez un sous-réseau public pour les ressources qui doivent être connectées à Internet et un sous-réseau privé pour les ressources qui ne doivent pas l'être connectées.
- Groupes de sécurité : un groupe de sécurité contrôle le trafic autorisé à atteindre et à quitter les ressources auxquelles il est associé. Les groupes de sécurité fournissent une couche de sécurité supplémentaire pour protéger les AWS ressources de chaque sous-réseau, ce qui vous permet de mieux contrôler le trafic réseau. Lorsque vous créez un VPC, celui-ci est fourni avec un groupe de sécurité par défaut. Vous pouvez créer des groupes de sécurité supplémentaires pour chaque VPC. Vous pouvez associer un groupe de sécurité uniquement aux ressources du VPC pour lequel il a été créé.
- Double pile : une double pile est un type d'adresse qui prend en charge le trafic réseau provenant à la fois des points de terminaison IPv4 et IPv6 des points de terminaison.

Terme spécifique à la configuration du trafic sortant

Connecteur VPC

Un connecteur VPC est une ressource App Runner qui permet au service App Runner d'accéder aux applications exécutées dans un Amazon VPC privé.

Termes spécifiques à la configuration du trafic entrant

Pour savoir comment rendre vos services accessibles de manière privée uniquement depuis un Amazon VPC, comprenons les termes suivants :

- Connexion d'entrée VPC : La connexion d'entrée VPC est une ressource App Runner qui fournit un point de terminaison App Runner pour le trafic entrant. App Runner attribue la ressource de connexion d'entrée VPC en arrière-plan lorsque vous choisissez un point de terminaison privé sur

la console App Runner pour votre trafic entrant. La ressource VPC Ingress Connection connecte votre service App Runner au point de terminaison de l'interface VPC d'Amazon VPC.

> **Note**
>
> Si vous utilisez l'API App Runner, la ressource VPC Ingress Connection n'est pas créée automatiquement.

- Point de terminaison privé : le point de terminaison privé est une option de console App Runner que vous sélectionnez pour configurer le trafic réseau entrant afin qu'il soit accessible uniquement depuis un Amazon VPC.

Configuration des configurations réseau pour le trafic entrant

Vous pouvez configurer votre service pour recevoir le trafic entrant depuis un point de terminaison privé ou public.

Un point de terminaison public est la configuration par défaut. Il ouvre votre service à tout trafic entrant en provenance de l'Internet public. Il vous offre également la possibilité de choisir entre des types d'adresses IPv4 ou à double pile (IPv4 et IPv6) pour votre service.

Un point de terminaison privé autorise uniquement le trafic provenant d'un Amazon VPC à accéder à votre service App Runner. Cela est possible en configurant un point de terminaison d'interface VPC, une AWS PrivateLink ressource, pour votre service App Runner. Ainsi, vous créez une connexion privée entre Amazon VPC et votre service App Runner. Il vous offre également la possibilité de choisir entre des types d'adresses IPv4 ou à double pile (IPv4 et IPv6) pour votre service.

Les sujets suivants sont abordés dans le cadre de la configuration de votre réseau pour le trafic entrant :

- Comment configurer votre trafic entrant pour que votre service soit disponible en privé uniquement depuis un Amazon VPC. Pour plus d'informations, consultez la section Activation du point de terminaison privé pour le trafic entrant.
- Comment configurer votre service pour recevoir du trafic Internet à partir du type d'adresse à double pile. Pour plus d'informations, consultez la section Activation du double stack pour le trafic entrant public.

En-têtes

Avec App Runner, vous pouvez accéder à la source IPv4 et IPv6 aux adresses d'origine du trafic entrant dans votre application. Les adresses IP sources d'origine sont préservées en leur attribuant l'en-tête de `X-Forwarded-For` demande. Cela permet à vos applications de récupérer les adresses IP sources d'origine en cas de besoin.

> **Note**
>
> Si votre service est configuré pour utiliser un point de terminaison privé, l'en-tête de `X-Forwarded-For` demande ne peut pas être utilisé pour accéder aux adresses IP source d'origine. S'il est utilisé, il récupère les fausses valeurs.

Activation du point de terminaison privé pour le trafic entrant

Par défaut, lorsque vous créez un AWS App Runner service, celui-ci est accessible via Internet. Cependant, vous pouvez également rendre votre service App Runner privé et uniquement accessible depuis un Amazon Virtual Private Cloud (Amazon VPC).

Lorsque votre service App Runner est privé, vous avez un contrôle total sur le trafic entrant, ce qui ajoute un niveau de sécurité supplémentaire. Cela est utile dans de nombreux cas d'utilisation, notamment pour exécuter des applications Web internes ou d'entreprise APIs, ou des applications encore en développement qui nécessitent un niveau de confidentialité et de sécurité plus élevé, ou qui doivent répondre à des exigences de conformité spécifiques.

> **Note**
>
> Si votre application App Runner nécessite des règles de contrôle du trafic IP/CIDR entrant à la source, vous devez utiliser des règles de groupe de sécurité pour les points de terminaison privés plutôt que pour le Web WAF. ACLs Cela est dû au fait que nous ne prenons actuellement pas en charge le transfert des données IP source des demandes vers les services privés App Runner associés à WAF. Par conséquent, les règles IP source pour les services privés App Runner associés au Web WAF ACLs ne respectent pas les règles basées sur l'IP.
> Pour en savoir plus sur la sécurité de l'infrastructure et les groupes de sécurité, notamment sur les meilleures pratiques, consultez les rubriques suivantes du guide de l'utilisateur

Amazon VPC : Contrôlez le trafic réseau et contrôlez le trafic vers vos ressources AWS à l'aide de groupes de sécurité.

Lorsque votre service App Runner est privé, vous pouvez y accéder depuis un Amazon VPC. Aucune passerelle Internet, périphérique NAT ou connexion VPN n'est requise.

> **Note**
>
> App Runner prend IPv4 en charge le double stack (IPv4 les deux IPv6) pour le trafic entrant et le trafic sortant.

Considérations

- Avant de configurer un point de terminaison d'interface VPC pour App Runner, consultez les considérations du AWS PrivateLink guide.
- Les politiques de point de terminaison VPC ne sont pas prises en charge pour App Runner. Par défaut, l'accès complet à App Runner est autorisé via le point de terminaison de l'interface VPC. Vous pouvez également associer un groupe de sécurité aux interfaces réseau du point de terminaison pour contrôler le trafic vers App Runner via le point de terminaison de l'interface VPC.
- Si votre application App Runner nécessite des règles de contrôle du trafic IP/CIDR entrant à la source, vous devez utiliser des règles de groupe de sécurité pour les points de terminaison privés plutôt que pour le Web WAF. ACLs Cela est dû au fait que nous ne prenons actuellement pas en charge le transfert des données IP source des demandes vers les services privés App Runner associés à WAF. Par conséquent, les règles IP source pour les services privés App Runner associés au Web WAF ACLs ne respectent pas les règles basées sur l'IP.
- Une fois que vous avez activé un point de terminaison privé, votre service n'est accessible que depuis votre VPC et n'est pas accessible depuis Internet.
- Pour une disponibilité accrue, il est recommandé de sélectionner au moins deux sous-réseaux différents dans la zone de disponibilité pour le point de terminaison de l'interface VPC. Nous vous déconseillons d'utiliser un seul sous-réseau.
- Si vous choisissez l'option double pile pour le type d'adresse IP, assurez-vous que vos sous-réseaux peuvent prendre en charge le trafic à double pile.
- Vous pouvez utiliser le même point de terminaison d'interface VPC pour accéder à plusieurs services App Runner dans un VPC.

Pour plus d'informations sur les termes utilisés dans cette section, voir Terminologie.

Permissions

Voici la liste des autorisations requises pour activer le point de terminaison privé :

- EC2 : CreateTags
- EC2 : CreateVpcEndpoint
- EC2 : ModifyVpcEndpoint
- EC2 : DeleteVpcEndpoints
- EC2 : DescribeSubnets
- EC2 : DescribeVpcEndpoints
- EC2 : DescribeVpcs

Point de terminaison de l'interface VPC

Un point de terminaison d'interface VPC est une AWS PrivateLinkressource qui connecte un Amazon VPC à un service de point de terminaison. Vous pouvez spécifier dans quel Amazon VPC vous souhaitez que votre service App Runner soit accessible en passant un point de terminaison d'interface VPC. Pour créer un point de terminaison d'interface VPC, spécifiez les éléments suivants :

- Le VPC Amazon pour activer la connectivité.
- Ajoutez des groupes de sécurité. Par défaut, un groupe de sécurité est attribué au point de terminaison de l'interface VPC. Vous pouvez choisir d'associer un groupe de sécurité personnalisé pour renforcer le contrôle du trafic réseau entrant.
- Ajoutez des sous-réseaux. Pour garantir une meilleure disponibilité, il est recommandé de sélectionner au moins deux sous-réseaux pour chaque zone de disponibilité à partir de laquelle vous allez accéder au service App Runner. Un point de terminaison d'interface réseau est créé dans chaque sous-réseau que vous activez pour le point de terminaison d'interface VPC. Il s'agit d'interfaces réseau gérées par les demandeurs qui servent de point d'entrée pour le trafic destiné à App Runner. Une interface réseau gérée par le demandeur est une interface réseau créée en votre nom par un AWS service dans votre VPC.
- Si vous utilisez l'API, ajoutez le point de terminaison de l'interface VPC App Runner. `Servicename` Par exemple,

```
com.amazonaws.region.apprunner.requests
```

Vous pouvez créer un point de terminaison d'interface VPC à l'aide de l'un des services suivants : AWS

- Console App Runner. Pour plus d'informations, consultez Gérer un point de terminaison privé.
- Console ou API Amazon VPC, et AWS Command Line Interface ()AWS CLI. Pour plus d'informations, consultez Accès aux Services AWS via AWS PrivateLink dans le Guide AWS PrivateLink .

> **Note**
>
> Vous êtes facturé pour chaque point de terminaison d'interface VPC que vous utilisez en fonction AWS PrivateLink de la tarification. Par conséquent, pour une meilleure rentabilité, vous pouvez utiliser le même point de terminaison d'interface VPC pour accéder à plusieurs services App Runner au sein d'un VPC. Toutefois, pour une meilleure isolation, pensez à associer un point de terminaison d'interface VPC différent pour chacun de vos services App Runner.

Connexion d'entrée de VPC

Une connexion d'entrée VPC est une ressource App Runner qui spécifie un point de terminaison App Runner pour le trafic entrant. App Runner attribue la ressource de connexion d'entrée VPC en arrière-plan lorsque vous choisissez un point de terminaison privé sur la console App Runner pour votre trafic entrant. Choisissez cette option pour autoriser uniquement le trafic provenant d'un Amazon VPC à accéder à votre service App Runner. La ressource VPC Ingress Connection connecte votre service App Runner au point de terminaison de l'interface VPC d'Amazon VPC. Vous pouvez créer une ressource de connexion d'entrée VPC uniquement si vous utilisez les opérations d'API pour configurer les paramètres réseau pour le trafic entrant. Pour plus d'informations sur la création d'une ressource de connexion d'entrée VPC, consultez la référence de CreateVpcIngressConnectionl'AWS App Runner API.

> **Note**
>
> Une ressource de connexion d'entrée VPC de l'App Runner peut se connecter à un point de terminaison d'interface VPC de l'Amazon VPC. En outre, vous ne pouvez créer qu'une seule ressource de connexion d'entrée VPC pour chaque service App Runner.

Point de terminaison privé

Le point de terminaison privé est une option de console App Runner que vous pouvez choisir si vous souhaitez uniquement recevoir du trafic entrant en provenance d'un Amazon VPC. Le choix de l'option Point de terminaison privé sur la console App Runner vous permet de connecter votre service à un VPC en configurant son point de terminaison d'interface VPC. Dans les coulisses, App Runner attribue une ressource de connexion d'entrée VPC au point de terminaison de l'interface VPC que vous configurez.

Résumé

Rendez votre service privé en autorisant uniquement le trafic provenant d'un Amazon VPC à accéder à votre service App Runner. Pour ce faire, vous créez un point de terminaison d'interface VPC pour le VPC Amazon sélectionné à l'aide d'App Runner ou d'Amazon VPC. Sur la console App Runner, vous créez un point de terminaison d'interface VPC lorsque vous activez le point de terminaison privé pour le trafic entrant. App Runner crée ensuite automatiquement une ressource de connexion d'entrée VPC et se connecte au point de terminaison de l'interface VPC et à votre service App Runner. Cela crée une connexion de service privée qui garantit que seul le trafic provenant du VPC sélectionné peut accéder à votre service App Runner.

Gestion des points de terminaison privés

Gérez le point de terminaison privé pour le trafic entrant à l'aide de l'une des méthodes suivantes :

- the section called “Console App Runner”
- the section called “API App Runner ou AWS CLI”

> **Note**
>
> Si votre application App Runner nécessite des règles de contrôle du trafic IP/CIDR entrant à la source, vous devez utiliser des règles de groupe de sécurité pour les points de terminaison privés plutôt que pour le Web WAF. ACLs Cela est dû au fait que nous ne prenons actuellement pas en charge le transfert des données IP source des demandes vers les services privés App Runner associés à WAF. Par conséquent, les règles IP source pour les services privés App Runner associés au Web WAF ACLs ne respectent pas les règles basées sur l'IP.
>
> Pour en savoir plus sur la sécurité de l'infrastructure et les groupes de sécurité, notamment sur les meilleures pratiques, consultez les rubriques suivantes du guide de l'utilisateur

Amazon VPC : Contrôlez le trafic réseau et contrôlez le trafic vers vos ressources AWS à l'aide de groupes de sécurité.

Console App Runner

Lorsque vous créez un service à l'aide de la console App Runner, ou lorsque vous mettez à jour sa configuration ultérieurement, vous pouvez choisir de configurer le trafic entrant.

Pour configurer votre trafic entrant, choisissez l'une des options suivantes.

- Point de terminaison public : pour rendre votre service accessible à tous les services via Internet. Par défaut, le point de terminaison public est sélectionné.
- Point de terminaison privé : pour rendre votre service App Runner accessible uniquement depuis un Amazon VPC.

Activer le point de terminaison privé

Activez un point de terminaison privé en l'associant au point de terminaison d'interface VPC de l'Amazon VPC auquel vous souhaitez accéder. Vous pouvez soit créer un nouveau point de terminaison d'interface VPC, soit en choisir un existant.

Pour créer un point de terminaison d'interface VPC

1. Ouvrez la console App Runner, puis dans la liste des régions, sélectionnez votre Région AWS.
2. Accédez à la section Réseau sous Configurer le service.
3. Choisissez Point de terminaison privé pour le trafic réseau entrant. Les options permettant de se connecter à un VCP à l'aide du point de terminaison de l'interface VPC s'ouvrent.
4. Choisissez Créer un nouveau point de terminaison. La boîte de dialogue Créer un nouveau point de terminaison d'interface VPC s'ouvre.
5. Entrez un nom pour le point de terminaison de l'interface VPC.
6. Choisissez le point de terminaison d'interface VPC requis dans la liste déroulante disponible.
7. Choisissez le groupe de sécurité dans la liste déroulante. L'ajout de groupes de sécurité fournit une couche de sécurité supplémentaire au point de terminaison de l'interface VPC. Il est recommandé de choisir au moins deux groupes de sécurité. Si vous ne choisissez aucun groupe de sécurité, App Runner attribue un groupe de sécurité par défaut au point de terminaison

de l'interface VPC. Assurez-vous que les règles du groupe de sécurité ne bloquent pas les ressources qui souhaitent communiquer avec votre service App Runner. Les règles du groupe de sécurité doivent autoriser les ressources qui interagiront avec votre service App Runner.

> **Note**
>
> Si votre application App Runner nécessite des règles de contrôle du trafic IP/CIDR entrant à la source, vous devez utiliser des règles de groupe de sécurité pour les points de terminaison privés plutôt que pour le Web WAF. ACLs Cela est dû au fait que nous ne prenons actuellement pas en charge le transfert des données IP source des demandes vers les services privés App Runner associés à WAF. Par conséquent, les règles IP source pour les services privés App Runner associés au Web WAF ACLs ne respectent pas les règles basées sur l'IP.
> Pour en savoir plus sur la sécurité de l'infrastructure et les groupes de sécurité, notamment sur les meilleures pratiques, consultez les rubriques suivantes du guide de l'utilisateur Amazon VPC : Contrôlez le trafic réseau et contrôlez le trafic vers vos ressources AWS à l'aide de groupes de sécurité.

8. Choisissez les sous-réseaux requis dans la liste déroulante. Il est recommandé de sélectionner au moins deux sous-réseaux pour chaque zone de disponibilité à partir de laquelle vous allez accéder au service App Runner.

> **Note**
>
> Si vous configurez le point de terminaison pour une double pile, assurez-vous que votre infrastructure et votre point de terminaison VPC prennent en charge le trafic à double pile.

9. (Facultatif) Choisissez Ajouter une nouvelle balise et entrez la clé de balise et la valeur de la balise.
10. Choisissez Créer. La page Configurer le service s'ouvre et affiche le message de création réussie du point de terminaison de l'interface VPC dans la barre supérieure.

Pour choisir un point de terminaison d'interface VPC existant

1. Ouvrez la console App Runner, puis dans la liste des régions, sélectionnez votre Région AWS.
2. Accédez à la section Réseau sous Configurer le service.

3. Choisissez Point de terminaison privé pour le trafic réseau entrant. Les options permettant de se connecter à un VPC à l'aide du point de terminaison de l'interface VPC s'ouvrent. La liste des points de terminaison d'interface VPC disponibles est affichée.
4. Choisissez le point de terminaison d'interface VPC requis répertorié sous Points de terminaison d'interface VPC.
5. Choisissez Next pour créer votre service. App Runner active le point de terminaison privé.

> **Note**
>
> Une fois votre service créé, vous pouvez choisir de modifier les groupes de sécurité et les sous-réseaux associés au point de terminaison de l'interface VPC, si nécessaire.

Pour vérifier les détails du point de terminaison privé, accédez à votre service et développez la section Mise en réseau sous l'onglet Configuration. Il affiche les détails du VPC et du point de terminaison de l'interface VPC associé au point de terminaison privé.

Mettre à jour le point de terminaison de l'interface VPC

Une fois votre service App Runner créé, vous pouvez modifier le point de terminaison de l'interface VPC associé au point de terminaison privé.

> **Note**
>
> Vous ne pouvez pas mettre à jour le nom du point de terminaison et les champs VPC.

Pour mettre à jour le point de terminaison de l'interface VPC

1. Ouvrez la console App Runner, puis dans la liste des régions, sélectionnez votre Région AWS.
2. Accédez à votre service et choisissez Configurations réseau dans le panneau de gauche.
3. Choisissez Trafic entrant pour afficher les points de terminaison de l'interface VPC associés aux services respectifs.
4. Choisissez le point de terminaison de l'interface VPC que vous souhaitez modifier.
5. Choisissez Modifier. La boîte de dialogue permettant de modifier le point de terminaison de l'interface VPC s'ouvre.

6. Choisissez les groupes de sécurité et les sous-réseaux requis, puis cliquez sur Mettre à jour. La page présentant les détails du point de terminaison de l'interface VPC s'ouvre avec le message de mise à jour réussie du point de terminaison de l'interface VPC dans la barre supérieure.

> **Note**
>
> Si votre application App Runner nécessite des règles de contrôle du trafic IP/CIDR entrant à la source, vous devez utiliser des règles de groupe de sécurité pour les points de terminaison privés plutôt que pour le Web WAF. ACLs Cela est dû au fait que nous ne prenons actuellement pas en charge le transfert des données IP source des demandes vers les services privés App Runner associés à WAF. Par conséquent, les règles IP source pour les services privés App Runner associés au Web WAF ACLs ne respectent pas les règles basées sur l'IP.
> Pour en savoir plus sur la sécurité de l'infrastructure et les groupes de sécurité, notamment sur les meilleures pratiques, consultez les rubriques suivantes du guide de l'utilisateur Amazon VPC : Contrôlez le trafic réseau et contrôlez le trafic vers vos ressources AWS à l'aide de groupes de sécurité.

Supprimer le point de terminaison de l'interface VPC

Si vous ne souhaitez pas que votre service App Runner soit accessible de manière privée, vous pouvez définir votre trafic entrant sur Public. Le passage à Public supprime le point de terminaison privé, mais pas le point de terminaison de l'interface VPC

Pour supprimer le point de terminaison de l'interface VPC

1. Ouvrez la console App Runner, puis dans la liste des régions, sélectionnez votre Région AWS.
2. Accédez à votre service et choisissez Configurations réseau dans le panneau de gauche.
3. Choisissez Trafic entrant pour afficher les points de terminaison de l'interface VPC associés aux services respectifs.

> **Note**
>
> Avant de supprimer un point de terminaison d'interface VPC, supprimez-le de tous les services auxquels il est connecté en mettant à jour votre service.

4. Sélectionnez Delete (Supprimer).

Si des services sont connectés au point de terminaison de l'interface VPC, vous recevez le message Impossible de supprimer le point de terminaison de l'interface VPC. Si aucun service n'est connecté au point de terminaison de l'interface VPC, vous recevez un message confirmant la suppression.

5. Sélectionnez Delete (Supprimer). La page des configurations réseau s'ouvre pour le trafic entrant avec le message de suppression réussie du point de terminaison de l'interface VPC dans la barre supérieure.

API App Runner ou AWS CLI

Vous pouvez déployer une application sur App Runner uniquement accessible depuis un Amazon VPC.

Pour plus d'informations sur les autorisations requises pour rendre votre service privé, consultezthe section called “Permissions”.

Pour créer une connexion de service privée à Amazon VPC

1. Créez un point de terminaison d'interface VPC, une AWS PrivateLink ressource, pour vous connecter à App Runner. Pour ce faire, spécifiez les sous-réseaux et les groupes de sécurité à associer à l'application. Voici un exemple de création d'un point de terminaison d'interface VPC.

> **Note**
>
> Si votre application App Runner nécessite des règles de contrôle du trafic IP/CIDR entrant à la source, vous devez utiliser des règles de groupe de sécurité pour les points de terminaison privés plutôt que pour le Web WAF. ACLs Cela est dû au fait que nous ne prenons actuellement pas en charge le transfert des données IP source des demandes vers les services privés App Runner associés à WAF. Par conséquent, les règles IP source pour les services privés App Runner associés au Web WAF ACLs ne respectent pas les règles basées sur l'IP.
> Pour en savoir plus sur la sécurité de l'infrastructure et les groupes de sécurité, notamment sur les meilleures pratiques, consultez les rubriques suivantes du guide de l'utilisateur Amazon VPC : Contrôlez le trafic réseau et contrôlez le trafic vers vos ressources AWS à l'aide de groupes de sécurité.

Example

```
aws ec2 create-vpc-endpoint
 --vpc-endpoint-type: Interface
 --service-name: com.amazonaws.us-east-1.apprunner.requests
 --subnets: subnet1, subnet2
 --security-groups: sg1
```

2. Référencez le point de terminaison de l'interface VPC en utilisant les actions de l'API CreateServiceou UpdateServiceApp Runner via la CLI. Configurez votre service pour qu'il ne soit pas accessible au public. `IsPubliclyAccessible`Défini `False` sur dans le `IngressConfiguration` membre du `NetworkConfiguration` paramètre. Vous pouvez éventuellement définir le `IpAddressType` champ sur `IPV4` ou`DUAL_STACK`. Si elle n'est pas définie, cette valeur par défaut est. IPV4 L'exemple suivant fait référence au point de terminaison de l'interface VPC.

   Example

   ```
   aws apprunner create-service
    --network-configuration:
      {
      "IngressConfiguration":
         {
         "IsPubliclyAccessible": False
         },
         "IpAddressType": "IPV4"
      }
    --service-name: com.amazonaws.us-east-1.apprunner.requests
    --source-configuration: <source_configuration>
   ```

3. Appelez l'action `create-vpc-ingress-connection` API pour créer la ressource de connexion d'entrée VPC pour App Runner et associez-la au point de terminaison de l'interface VPC que vous avez créé à l'étape précédente. Il renvoie un nom de domaine qui est utilisé pour accéder à votre service dans le VPC spécifié. Voici un exemple de création d'une ressource de connexion d'entrée VPC.

   Example Demande

   ```
   aws apprunner create-vpc-ingress-connection
   ```

```
 --service-arn: <apprunner_service_arn>
 --ingress-vpc-configuration: {"VpcId":<vpc_id>, "VpceId": <vpce_id>}
 --vpc-ingress-connection-name: <vic_connection_name>
```

Example Réponse

```
{
   "VpcIngressConnectionArn": <vpc_ingress_connection_arn>,
   "VpcIngressConnectionName": <vic_connection_name>,
   "ServiceArn": <apprunner_service_arn>,
   "Status": "PENDING_CREATION",
   "AccountId": <connection_owner_id>,
   "DomainName": <domain_name_associated_with_vpce>,
   "IngressVpcConfiguration": {"VpcId":<vpc_id>, "VpceId":<vpce_id>},
   "CreatedAt": <date_created>
}
```

Mettre à jour la connexion d'entrée VPC

Vous pouvez mettre à jour la ressource VPC Ingress Connection. La connexion d'entrée VPC doit être dans l'un des états suivants pour être mise à jour :

- DISPONIBLE
- ÉCHEC DE LA CRÉATION
- ÉCHEC DE LA MISE À JOUR

Voici un exemple de mise à jour d'une ressource de connexion d'entrée VPC.

Example Demande

```
aws apprunner update-vpc-ingress-connection
      --vpc-ingress-connection-arn: <vpc_ingress_connection_arn>
```

Example Réponse

```
{
    "VpcIngressConnectionArn":  <vpc_ingress_connection_arn>,
    "VpcIngressConnectionName":  <vic_connection_name>,
    "ServiceArn":  <apprunner_service_arn>,
```

```
    "Status": "FAILED_UPDATE",
    "AccountId":  <connection_owner_id>,
    "DomainName":  <domain_name_associated_with_vpce>,
    "IngressVpcConfiguration": {"VpcId":<vpc_id>, "VpceId":<vpce_id>},
    "CreatedAt":  <date_created>
}
```

Supprimer la connexion d'entrée VPC

Vous pouvez supprimer la ressource VPC Ingress Connection si vous n'avez plus besoin de la connexion privée à Amazon VPC.

La connexion d'entrée VPC doit être dans l'un des états suivants pour être supprimée :

- DISPONIBLE
- ÉCHEC DE LA CRÉATION
- ÉCHEC DE MISE À JOUR
- ÉCHEC DE LA SUPPRESSION

Voici un exemple de suppression d'une connexion d'entrée VPC

Example Demande

```
aws apprunner delete-vpc-ingress-connection
      --vpc-ingress-connection-arn: <vpc_ingress_connection_arn>
```

Example Réponse

```
{
   "VpcIngressConnectionArn": <vpc_ingress_connection_arn>,
   "VpcIngressConnectionName": <vic_connection_name>,
   "ServiceArn": <apprunner_service_arn>,
   "Status": "PENDING_DELETION",
   "AccountId": <connection_owner_id>,
   "DomainName": <domain_name_associated_with_vpce>,
   "IngressVpcConfiguration": {"VpcId":<vpc_id>, "VpceId":<vpce_id>},
   "CreatedAt": <date_created>,
   "DeletedAt": <date_deleted>
}
```

Utilisez les actions d'API App Runner suivantes pour gérer le trafic entrant privé pour votre service.

- CreateVpcIngressConnection— Créez une nouvelle ressource de connexion d'entrée VPC. App Runner a besoin de cette ressource lorsque vous souhaitez associer votre service App Runner à un point de terminaison Amazon VPC.
- ListVpcIngressConnections— Renvoie une liste des points de terminaison de connexion AWS App Runner VPC Ingress associés à votre compte. AWS
- DescribeVpcIngressConnection— Renvoie une description complète de la ressource AWS App Runner VPC Ingress Connection.
- UpdateVpcIngressConnection— Mettez à jour la ressource AWS App Runner VPC Ingress Connection.
- DeleteVpcIngressConnection— Supprimez une ressource de connexion d'entrée VPC App Runner associée au service App Runner.

Pour plus d'informations sur l'utilisation de l'API App Runner, consultez le guide de référence de l'API App Runner.

IPv6 Activation du trafic entrant

Si vous souhaitez que votre service reçoive le trafic réseau entrant depuis des IPv6 adresses, ou depuis les deux IPv4 IPv6 adresses, choisissez le type d'adresse à double pile pour le point de terminaison. Lorsque vous créez une nouvelle application, vous pouvez trouver ce paramètre dans la section Configurer le service > Mise en réseau. Les procédures suivantes expliquent comment activer IPv4 ou dupliquer (IPv6 et IPv4) à l'aide de la console App Runner ou de l'API App Runner.

Gestion du double stack pour le trafic entrant

Gérez le type d'adresse à double pile pour le trafic entrant à l'aide de l'une des méthodes suivantes :

- the section called “Console App Runner”
- the section called “API App Runner ou AWS CLI”

> **Note**
>
> Les procédures suivantes expliquent comment gérer le type d'adresse réseau pour le trafic public entrant. Pour plus d'informations sur la gestion des types d' IPv4 adresses ou à double

pile pour les points de terminaison privés, consultez. the section called “Gérer les points de terminaison privés”

Console App Runner

Vous pouvez choisir le type d'adresse à double pile pour le trafic Internet entrant, lorsque vous créez un service à l'aide de la console App Runner ou lorsque vous mettez à jour sa configuration ultérieurement.

Pour activer le type d'adresse à double pile

1. Lorsque vous créez ou mettez à jour un service, développez la section Réseau sous Configurer le service.
2. Choisissez Public Endpoint pour le trafic réseau entrant. Si vous sélectionnez Point de terminaison public, l'option Type d'adresse IP du point de terminaison s'ouvre.

   Consultez the section called “Gérer les points de terminaison privés” la procédure à suivre pour gérer les types d' IPv4adresses ou à double pile pour les points de terminaison privés.
3. Développez le type d'adresse IP du point de terminaison pour afficher les types d'adresses IP suivants.

   - IPv4
   - Double pile (IPv4 et IPv6)

   > **Note**
   >
   > Si vous n'étendez pas le type d'adresse IP du point de terminaison pour effectuer une sélection, App Runner l'attribue IPv4 comme configuration par défaut.
4. Choisissez Dual-stack (IPv4 et IPv6).
5. Choisissez Next, puis Create & Deploy si vous créez un service. Sinon, choisissez Enregistrer les modifications si vous mettez à jour un service.

   Lorsque le service est déployé, votre application commence à recevoir du trafic réseau provenant à la fois des IPv6 terminaux IPv4 et des terminaux.

Pour modifier le type d'adresse

1. Suivez les étapes pour mettre à jour un service et accédez à Networking.
2. Accédez au type d'adresse IP du point de terminaison sous Trafic réseau entrant et sélectionnez le type d'adresse requis.
3. Sélectionnez Enregistrer les modifications. Votre service est mis à jour avec votre sélection.

API App Runner ou AWS CLI

Lorsque vous appelez les actions CreateServiceou UpdateServicel'API App Runner, utilisez le `IpAddressType` membre du `NetworkConfiguration` paramètre pour spécifier le type d'adresse. Les valeurs prises en charge que vous pouvez spécifier sont `IPv4` et`DUAL_STACK`. Spécifiez `DUAL_STACK` si vous souhaitez que votre service reçoive du trafic Internet en provenance IPv4 et des IPv6 points de terminaison. Si vous ne spécifiez aucune valeur pour`IpAddressType`, elle IPv4 est appliquée par défaut.

> **Note**
>
> Pour des exemples de points de terminaison privés, voirthe section called “API App Runner ou AWS CLI”.

Voici l'exemple de création d'un service avec la double pile comme adresse IP. Cet exemple appelle un `input.json` fichier.

Example Demande de création d'un service avec support Dual Stack

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

Example Contenu de **`input.json`**

```
{
  "ServiceName": "example-service",
  "SourceConfiguration": {
    "ImageRepository": {
      "ImageIdentifier": "public.ecr.aws/aws-containers/hello-app-runner:latest",
      "ImageConfiguration": {
        "Port": "8000"
```

```
      },
      "ImageRepositoryType": "ECR_PUBLIC"
    },
    "NetworkConfiguration": {
      "IpAddressType": "DUAL_STACK"
    }
  }
}
```

Example Réponse

```
{
  "Service": {
    "ServiceName": "example-service",
    "ServiceId": "<service-id>",
    "ServiceArn": "arn:aws:apprunner:us-east-2:123456789012:service/example-
service/<service-id>",
    "ServiceUrl": "1234567890.us-east-2.awsapprunner.com",
    "CreatedAt": "2023-10-16T12:30:51.724000-04:00",
    "UpdatedAt": "2023-10-16T12:30:51.724000-04:00",
    "Status": "OPERATION_IN_PROGRESS",
    "SourceConfiguration": {
      "ImageRepository": {
        "ImageIdentifier": "public.ecr.aws/aws-containers/hello-app-runner:latest",
        "ImageConfiguration": {
          "Port": "8000"
        },
        "ImageRepositoryType": "ECR_PUBLIC"
      },
      "AutoDeploymentsEnabled": false
    },
    "InstanceConfiguration": {
      "Cpu": "1024",
      "Memory": "2048"
    },
    "HealthCheckConfiguration": {
      "Protocol": "TCP",
      "Path": "/",
      "Interval": 5,
      "Timeout": 2,
      "HealthyThreshold": 1,
      "UnhealthyThreshold": 5
    },
```

```
    "AutoScalingConfigurationSummary": {
      "AutoScalingConfigurationArn": "arn:aws:apprunner:us-
east-2:123456789012:autoscalingconfiguration/
DefaultConfiguration/1/00000000000000000000000000000001",
      "AutoScalingConfigurationName": "DefaultConfiguration",
      "AutoScalingConfigurationRevision": 1
    },
    "NetworkConfiguration": {
      "IpAddressType": "DUAL_STACK",
      "EgressConfiguration": {
        "EgressType": "DEFAULT"
      },
      "IngressConfiguration": {
        "IsPubliclyAccessible": true
      }
    }
  },
  "OperationId": "24bd100b1e111ae1a1f0e1115c4f11de"
}
```

Pour plus d'informations sur le paramètre d'API, consultez NetworkConfiguration.

Activation de l'accès VPC pour le trafic sortant

Par défaut, votre AWS App Runner application peut envoyer des messages aux points de terminaison publics. Cela inclut vos propres solutions et tout autre site Web ou service Web public. Services AWS Votre application peut même envoyer des messages aux points de terminaison publics des applications qui s'exécutent dans un VPC depuis Amazon Virtual Private Cloud (Amazon VPC). Si vous ne configurez pas de VPC lorsque vous lancez votre environnement, App Runner utilise le VPC par défaut, qui est public.

Vous pouvez choisir de lancer votre environnement dans un VPC personnalisé afin de personnaliser les paramètres réseau et de sécurité pour le trafic sortant. Vous pouvez activer votre AWS App Runner service pour accéder aux applications qui s'exécutent dans un VPC privé depuis Amazon Virtual Private Cloud (Amazon VPC). Ensuite, votre application peut se connecter et envoyer des messages à d'autres applications hébergées dans un Amazon Virtual Private Cloud (Amazon VPC). Il s'agit par exemple d'une base de données Amazon RDS ElastiCache, d'Amazon et d'autres services privés hébergés dans un VPC privé.

Connecteur VPC

Vous pouvez associer votre service à un VPC en créant un point de terminaison VPC à partir de la console App Runner, appelé connecteur VPC. Pour créer un connecteur VPC, spécifiez le VPC, un ou plusieurs sous-réseaux et éventuellement un ou plusieurs groupes de sécurité. Après avoir configuré un connecteur VPC, vous pouvez l'utiliser avec un ou plusieurs services App Runner.

Latence unique

Si vous configurez votre service App Runner avec un connecteur VPC personnalisé pour le trafic sortant, il peut connaître une latence de démarrage unique de deux à cinq minutes. Le processus de démarrage attend que le connecteur VPC soit prêt à se connecter à d'autres ressources avant de définir l'état du service sur En cours d'exécution. Vous pouvez configurer un service avec un connecteur VPC personnalisé lorsque vous le créez pour la première fois, ou vous pouvez le faire ultérieurement en effectuant une mise à jour du service.

Notez que si vous réutilisez la même configuration de connecteur VPC pour un autre service, il n'y aura aucune latence. La configuration du connecteur VPC est basée sur la combinaison du groupe de sécurité et du sous-réseau. Pour une configuration de connecteur VPC donnée, la latence ne se produit qu'une seule fois, lors de la création initiale de l'hyperplan du connecteur VPC ENIs (interfaces réseau élastiques).

En savoir plus sur les connecteurs VPC personnalisés et Hyperplane AWS

Les connecteurs VPC d'App Runner sont basés sur AWS Hyperplane, le système réseau interne d'Amazon à l'origine de plusieurs AWS ressources, telles que Network Load Balancer, NAT Gateway et AWS. PrivateLink La technologie AWS Hyperplane fournit des capacités de débit élevé et de faible latence, ainsi qu'un degré de partage plus élevé. Un ENI Hyperplane est créé dans vos sous-réseaux lorsque vous créez un connecteur VPC et que vous l'associez à votre service. La configuration d'un connecteur VPC est basée sur une combinaison de groupe de sécurité et de sous-réseau, et vous pouvez référencer le même connecteur VPC sur plusieurs services App Runner. Par conséquent, l'hyperplan ENIs sous-jacent est partagé entre vos services App Runner. Ce partage est possible, même si vous augmentez le nombre de tâches requises pour gérer la charge de demandes, et permet une utilisation plus efficace de l'espace IP de votre VPC. Pour plus d'informations, consultez la section Présentation détaillée de la mise en réseau VPC d'AWS App Runner sur le blog AWS Container.

Sous-réseau

Chaque sous-réseau se trouve dans une zone de disponibilité spécifique. Pour une haute disponibilité, nous vous recommandons de sélectionner des sous-réseaux dans au moins trois zones de disponibilité. Si la région compte moins de trois zones de disponibilité, nous vous recommandons de sélectionner vos sous-réseaux dans toutes les zones de disponibilité prises en charge.

Lorsque vous sélectionnez un sous-réseau pour votre VPC, assurez-vous de choisir un sous-réseau privé et non un sous-réseau public. En effet, lorsque vous créez un connecteur VPC, le service App Runner crée une ENI Hyperplane dans chacun des sous-réseaux. Chaque ENI Hyperplane se voit attribuer une adresse IP privée uniquement et est étiquetée avec une étiquette de AWSAppRunnerManagedclé. Si vous choisissez un sous-réseau public, des erreurs se produiront lors de l'exécution de votre service App Runner. Toutefois, si votre service doit accéder à certains services disponibles sur Internet ou accessibles au public Services AWS, consultezthe section called “Considérations relatives à la sélection d'un sous-réseau”.

Considérations relatives à la sélection d'un sous-réseau

- Lorsque vous connectez votre service à un VPC, le trafic sortant n'a pas accès à l'Internet public. Tout le trafic sortant de votre application est dirigé via le VPC auquel votre service est connecté. Toutes les règles de mise en réseau du VPC s'appliquent au trafic sortant de votre application. Cela signifie que vos services ne peuvent pas accéder à l'Internet public et AWS APIs. Pour y accéder, effectuez l'une des opérations suivantes :
  - Connectez les sous-réseaux à Internet via une passerelle NAT.
  - Configurez les points de terminaison VPC Services AWS auxquels vous souhaitez accéder. Votre service reste dans le cadre d'Amazon VPC en utilisant. AWS PrivateLink
- Certaines zones de disponibilité Régions AWS ne prennent pas en charge les sous-réseaux qui peuvent être utilisés avec les services App Runner. Si vous choisissez des sous-réseaux dans ces zones de disponibilité, votre service ne sera ni créé ni mis à jour. Dans ces situations, App Runner fournit un message d'erreur détaillé pointant vers les sous-réseaux et les zones de disponibilité non pris en charge. Dans ce cas, résolvez le problème en supprimant les sous-réseaux non pris en charge de votre demande, puis réessayez.
- Les sous-réseaux que vous sélectionnez doivent tous avoir le même type d'adresse IP, qu'il s'agisse d'une double pile IPv4 ou d'une double pile.

Groupe de sécurité

Vous pouvez éventuellement spécifier les groupes de sécurité auxquels App Runner accède AWS sous les sous-réseaux spécifiés. Si vous ne spécifiez aucun groupe de sécurité, App Runner utilise le groupe de sécurité par défaut du VPC. Le groupe de sécurité par défaut autorise tout le trafic sortant.

L'ajout d'un groupe de sécurité fournit une couche de sécurité supplémentaire aux connecteurs VPC, vous permettant ainsi de mieux contrôler le trafic réseau. Le connecteur VPC est uniquement utilisé pour les communications sortantes depuis votre application. Vous utilisez des règles de sortie pour autoriser les communications vers les points de terminaison de destination souhaités. Vous devez également vous assurer que tous les groupes de sécurité associés à la ressource de destination disposent des règles entrantes appropriées. Dans le cas contraire, ces ressources ne peuvent pas accepter le trafic provenant des groupes de sécurité du connecteur VPC.

> **Note**
>
> Lorsque vous associez votre service à un VPC, le trafic suivant n'est pas affecté :
>
> - Trafic entrant : les messages entrants reçus par votre application ne sont pas affectés par un VPC associé. Les messages sont acheminés via le nom de domaine public associé à votre service et n'interagissent pas avec le VPC.
> - Trafic d'App Runner : App Runner gère plusieurs actions en votre nom, telles que l'extraction du code source et des images, le transfert de journaux et la récupération de secrets. Le trafic généré par ces actions n'est pas acheminé via votre VPC.

Pour en savoir plus sur le mode d' AWS App Runner intégration avec Amazon VPC, consultez AWS App Runner VPC Networking.

Gérer l'accès au VPC

> **Note**
>
> Si vous créez un connecteur VPC pour le trafic sortant pour un service, le processus de démarrage du service qui suit connaîtra une latence unique. Vous pouvez définir cette configuration pour un nouveau service lorsque vous le créez, ou ultérieurement, avec une mise à jour du service. Pour plus d'informations, consultez le Latence unique chapitre de ce guide consacré à la mise en réseau avec App Runner.

Gérez l'accès VPC pour vos services App Runner à l'aide de l'une des méthodes suivantes :

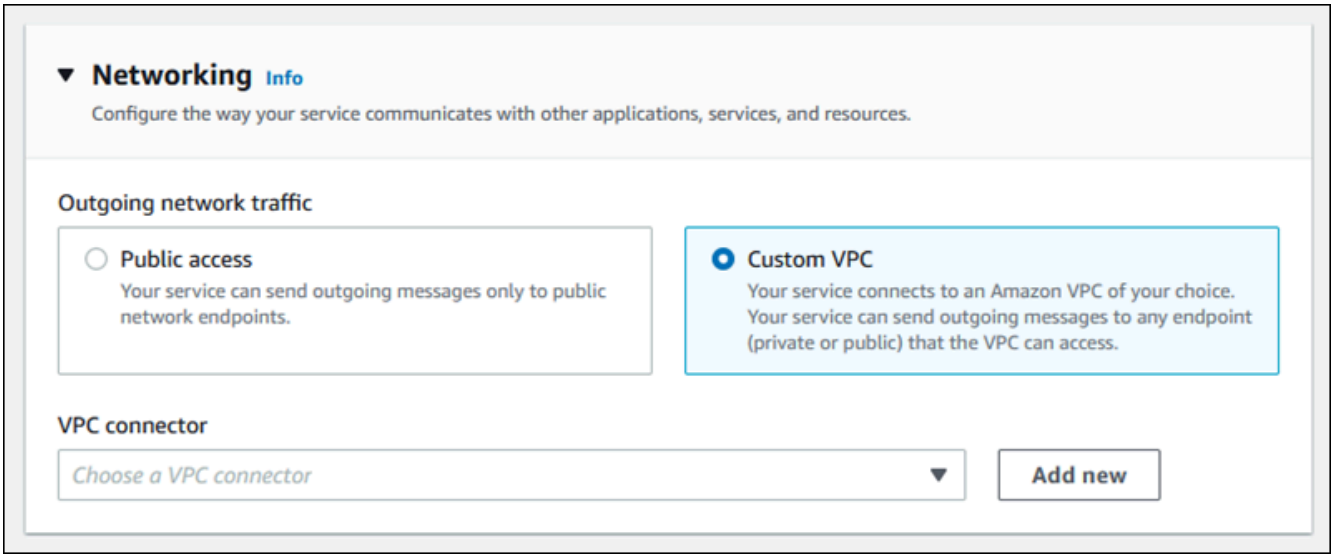
App Runner console

Lorsque vous créez un service à l'aide de la console App Runner, ou lorsque vous mettez à jour sa configuration ultérieurement, vous pouvez choisir de configurer votre trafic sortant. Consultez la section Configuration réseau sur la page de la console. Pour le trafic réseau sortant, choisissez l'une des options suivantes :

- Accès public : pour associer votre service à des points de terminaison publics ou à d'autres Services AWS.
- VPC personnalisé : pour associer votre service à un VPC d'Amazon VPC. Votre application peut se connecter et envoyer des messages à d'autres applications hébergées dans un Amazon VPC.

Pour activer le VPC personnalisé

1. Ouvrez la console App Runner, puis dans la liste des régions, sélectionnez votre Région AWS.
2. Accédez à la section Réseau sous Configurer le service.



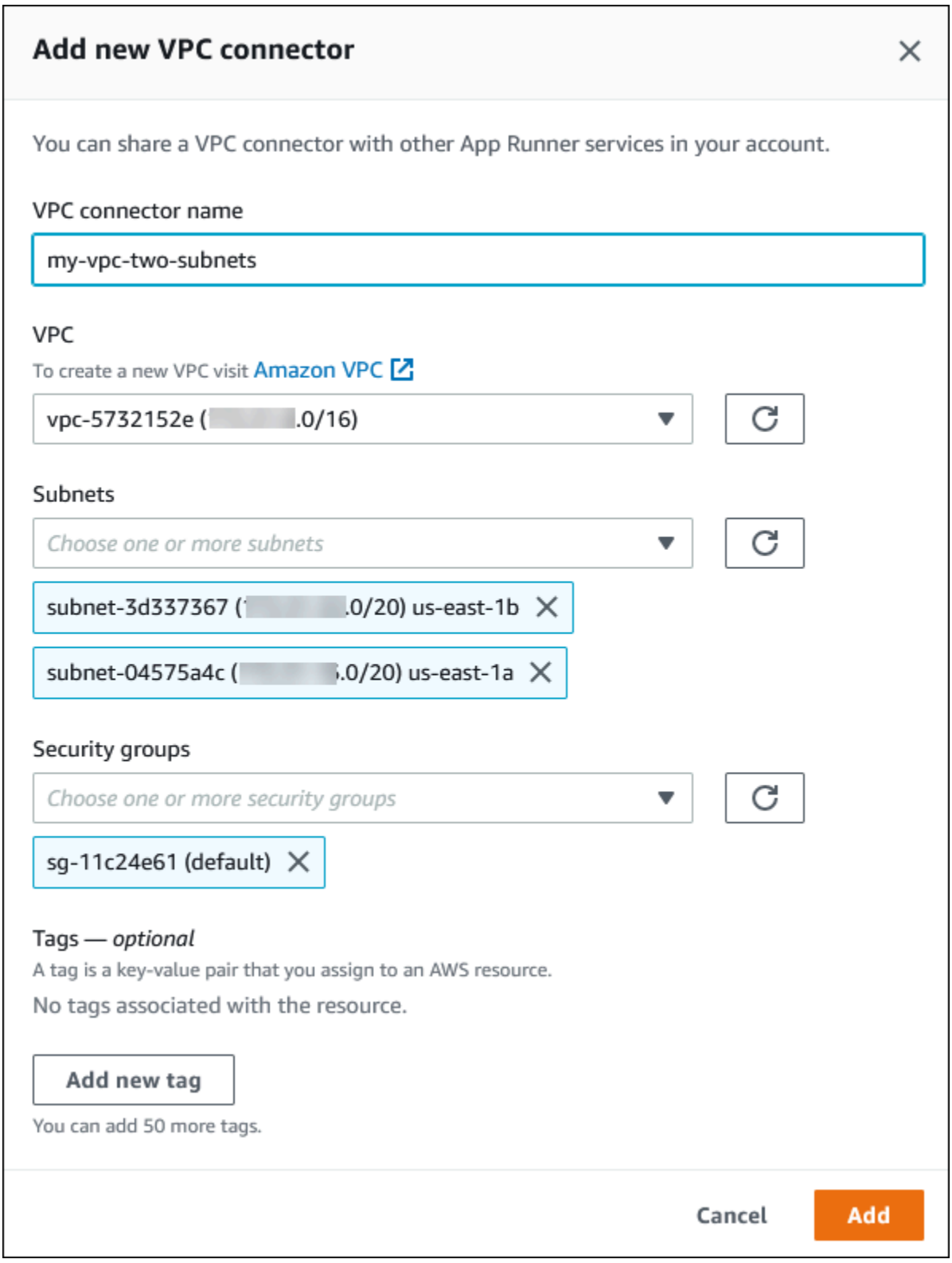
3. Choisissez VPC personnalisé, pour le trafic réseau sortant.
4. Dans le volet de navigation, choisissez le connecteur VPC.

   Si vous avez créé les connecteurs VPC, la console affiche la liste des connecteurs VPC de votre compte. Vous pouvez choisir un connecteur VPC existant et choisir Next pour revoir

votre configuration. Passez ensuite à la dernière étape. Vous pouvez également ajouter un nouveau connecteur VPC en procédant comme suit.

5. Choisissez Ajouter pour créer un nouveau connecteur VPC pour votre service.

   La boîte de dialogue Ajouter un nouveau connecteur VPC s'ouvre ensuite.



6. Entrez un nom pour votre connecteur VPC et sélectionnez le VPC requis dans la liste disponible.

7. Pour les sous-réseaux, sélectionnez un sous-réseau pour chaque zone de disponibilité à partir de laquelle vous prévoyez d'accéder au service App Runner. Pour une meilleure disponibilité, choisissez trois sous-réseaux. Ou, s'il existe moins de trois sous-réseaux, choisissez tous les sous-réseaux disponibles.

   > **Note**
   >
   > - Assurez-vous d'attribuer des sous-réseaux privés au connecteur VPC. Si vous attribuez des sous-réseaux publics au connecteur VPC, votre service ne parvient pas à être créé ou est annulé automatiquement lors d'une mise à jour.
   > - Si votre trafic sortant est à double pile, assurez-vous que tous les sous-réseaux que vous sélectionnez sont configurés pour une double pile dans la console VPC.

8. (Facultatif) Pour Groupe de sécurité, sélectionnez les groupes de sécurité à associer aux interfaces réseau des terminaux.
9. (Facultatif) Pour ajouter une balise, choisissez Ajouter une nouvelle balise et entrez la clé et la valeur de la balise.
10. Choisissez Ajouter.

    Les détails du connecteur VPC que vous avez créé apparaissent sous Connecteur VPC.
11. Choisissez Suivant pour vérifier votre configuration, puis choisissez Créer et déployer.

    App Runner crée une ressource de connecteur VPC pour vous, puis l'associe à votre service. Si le service est créé avec succès, la console affiche le tableau de bord du service, avec un aperçu du nouveau service.

App Runner API or AWS CLI

Lorsque vous appelez les actions CreateServiceou UpdateServicel'API App Runner, utilisez le `EgressConfiguration` membre du `NetworkConfiguration` paramètre pour spécifier une ressource de connecteur VPC pour votre service.

Utilisez les actions d'API App Runner suivantes pour gérer les ressources de votre connecteur VPC.

- CreateVpcConnector— Crée un nouveau connecteur VPC.

- ListVpcConnectors— Renvoie la liste des connecteurs VPC associés à votre. Compte AWS La liste inclut des descriptions complètes.
- DescribeVpcConnector— Renvoie une description complète d'un connecteur VPC.
- DeleteVpcConnector— Supprime un connecteur VPC. Si vous atteignez le quota de connecteurs VPC pour votre Compte AWS, vous devrez peut-être supprimer des connecteurs VPC inutiles.

Pour déployer une application sur App Runner disposant d'un accès sortant à un VPC, vous devez d'abord créer un connecteur VPC. Vous pouvez le faire en spécifiant un ou plusieurs sous-réseaux et groupes de sécurité à associer à l'application. Vous pouvez ensuite référencer le connecteur VPC dans Create ou UpdateServicevia la CLI, comme illustré dans l'exemple suivant :

```
            cat > vpc-connector.json <<EOF
{
"VpcConnectorName": "my-vpc-connector",
"Subnets": [
"subnet-a",
"subnet-b",
"subnet-c"
],
"SecurityGroups": [
"sg-1",
"sg-2"
]
}
EOF

aws apprunner create-vpc-connector \
--cli-input-json file:///vpc-connector.json

cat > service.json <<EOF

{
"ServiceName": "my-vpc-connected-service",
"SourceConfiguration": {
"ImageRepository": {
"ImageIdentifier": "<ecr-image-identifier> ",
"ImageConfiguration": {
"Port": "8000"
},
```

```
"ImageRepositoryType": "ECR"
}
},
"NetworkConfiguration": {
"EgressConfiguration": {
"EgressType": "VPC",
"VpcConnectorArn": "arn:aws:apprunner:..../my-vpc-connector"
}
}
}
EOF

aws apprunner create-service \
--cli-input-json file:///service.js
```

Observabilité pour votre service App Runner

AWS App Runner s'intègre à plusieurs AWS services afin de vous fournir une suite complète d'outils d'observabilité pour votre service App Runner. Les rubriques de ce chapitre décrivent ces fonctionnalités.

Rubriques

- Suivi de l'activité du service App Runner
- Afficher les logs d'App Runner transmis à CloudWatch Logs
- Affichage des statistiques de service App Runner signalées à CloudWatch
- Gestion des événements App Runner dans EventBridge
- Journalisation des appels d'API App Runner avec AWS CloudTrail
- Suivi de votre application App Runner avec X-Ray

Suivi de l'activité du service App Runner

AWS App Runner utilise une liste d'opérations pour suivre l'activité de votre service App Runner. Une opération représente un appel asynchrone à une action d'API, telle que la création d'un service, la mise à jour d'une configuration et le déploiement d'un service. Les sections suivantes expliquent comment suivre l'activité dans la console App Runner et à l'aide de l'API.

Suivez l'activité du service App Runner

Suivez l'activité de votre service App Runner à l'aide de l'une des méthodes suivantes :

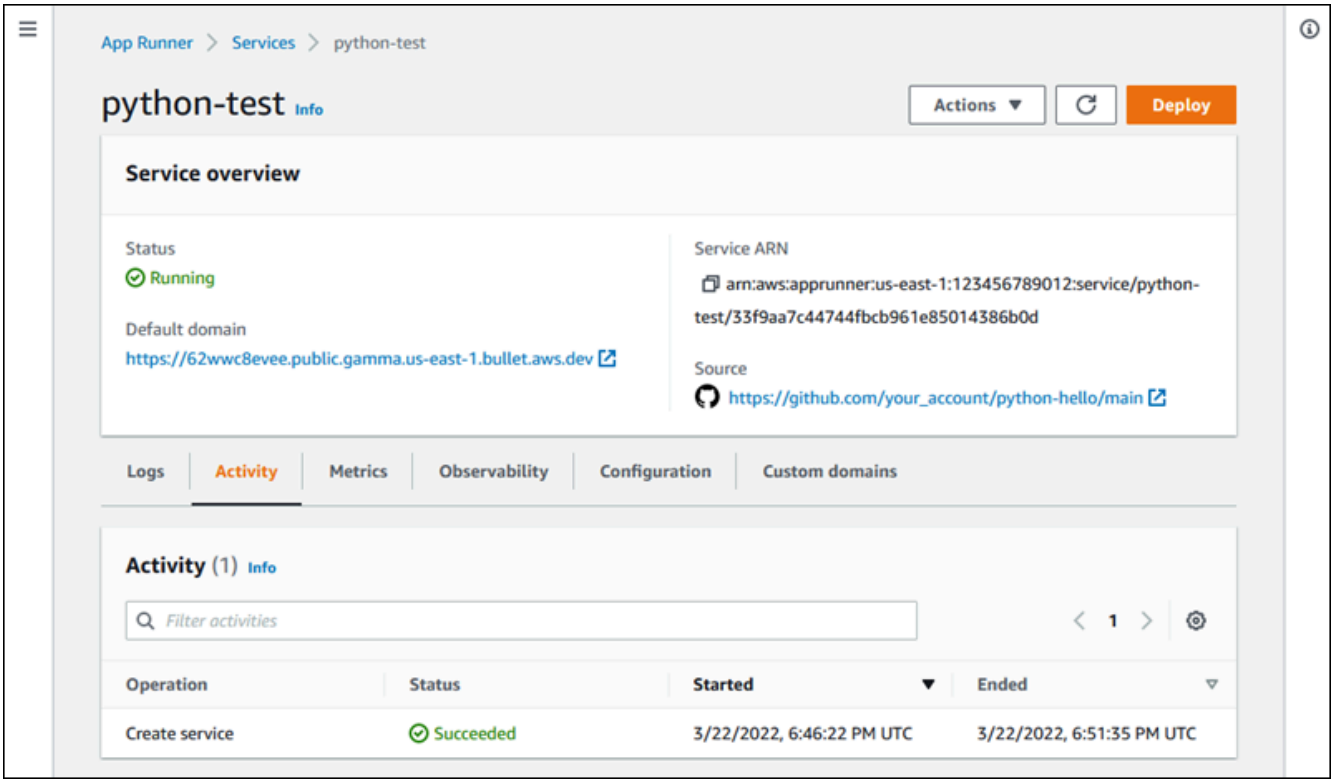
App Runner console

La console App Runner affiche l'activité de votre service App Runner et propose d'autres moyens d'explorer les opérations.

Pour consulter l'activité de votre service

1. Ouvrez la console App Runner, puis dans la liste des régions, sélectionnez votre Région AWS.
2. Dans le volet de navigation, choisissez Services, puis choisissez votre service App Runner.

   La console affiche le tableau de bord des services avec une vue d'ensemble des services.



3. Sur la page du tableau de bord du service, choisissez l'onglet Activité, s'il n'est pas déjà sélectionné.

   La console affiche la liste des opérations.

4. Pour rechercher des opérations spécifiques, descendez la liste en saisissant un terme de recherche. Vous pouvez rechercher n'importe quelle valeur figurant dans le tableau.

5. Choisissez une opération répertoriée pour voir ou télécharger le journal correspondant.

App Runner API or AWS CLI

L'ListOperationsaction, en fonction de l'Amazon Resource Name (ARN) d'un service App Runner, renvoie une liste des opérations effectuées sur ce service. Chaque élément de la liste contient un identifiant d'opération et des informations de suivi.

Afficher les logs d'App Runner transmis à CloudWatch Logs

Vous pouvez utiliser Amazon CloudWatch Logs pour surveiller, stocker et accéder aux fichiers journaux générés par vos ressources dans divers AWS services. Pour plus d'informations, consultez le guide de l'utilisateur d'Amazon CloudWatch Logs.

AWS App Runner collecte les résultats de vos déploiements d'applications et de votre service actif et les diffuse vers CloudWatch Logs. Les sections suivantes répertorient les flux de log d'App Runner et vous montrent comment les consulter dans la console App Runner.

Groupes de logs et flux App Runner

CloudWatch Les journaux conservent les données des journaux dans des flux de journaux qu'ils organisent ensuite en groupes de journaux. Un flux de journal est une séquence d'événements de journal provenant d'une source spécifique. Un groupe de journaux est un groupe de flux de journaux qui partagent les mêmes paramètres de conservation, de surveillance et de contrôle d’accès.

App Runner définit deux groupes de CloudWatch journaux, chacun avec plusieurs flux de journaux, pour chaque service App Runner de votre Compte AWS.

Journaux de service

Le groupe de journaux de service contient les résultats de journalisation générés par App Runner lorsqu'il gère votre service App Runner et agit en conséquence.

Nom du groupe de journaux	Exemple
`/aws/apprunner/` *`service-name`* `/` *`service-id`* `/service`	`/aws/apprunner/python-test/ac7ec8b51ff34746bcb6654e0bcb23da/service`

Au sein du groupe de journaux de service, App Runner crée un flux de journaux d'événements pour capturer l'activité pendant le cycle de vie de votre service App Runner. Il peut s'agir, par exemple, du lancement ou de la suspension de votre application.

En outre, App Runner crée un flux de journal pour chaque opération asynchrone de longue durée liée à votre service. Le nom du flux de journal reflète le type d'opération et l'ID d'opération spécifique.

Un déploiement est un type d'opération. Les journaux de déploiement contiennent les résultats de journalisation des étapes de création et de déploiement effectuées par App Runner lorsque vous créez un service ou déployez une nouvelle version de votre application. Les noms des flux du journal de déploiement commencent `deployment/` et se terminent par l'ID de l'opération qui effectue le déploiement. Cette opération est soit un CreateServiceappel pour le déploiement initial de l'application, soit un StartDeploymentappel pour chaque déploiement ultérieur.

Dans un journal de déploiement, chaque message de journal commence par un préfixe :

- `[AppRunner]`— Sortie générée par App Runner lors du déploiement.
- `[Build]`— Sortie de vos propres scripts de construction.

Nom du flux de log	Exemple
`events`	N/A (nom fixe)
operation-type `/`*operation-id*	`deployment/c2c8eeedea164f45 9cf78f12a8953390`

Journaux d'application

Le groupe de journaux d'applications contient la sortie du code de votre application en cours d'exécution.

Nom du groupe de journaux	Exemple
`/aws/apprunner/` *service-name* `/`*service-id* `/application`	`/aws/apprunner/python-test/ ac7ec8b51ff34746bcb6654e0bcb23da/ application`

Au sein du groupe de journaux d'applications, App Runner crée un flux de journal pour chaque instance (unité de dimensionnement) qui exécute votre application.

Nom du flux de log	Exemple
`instance/` *instance-id*	`instance/1a80bc9134a84699b7 b3432ebeebb591`

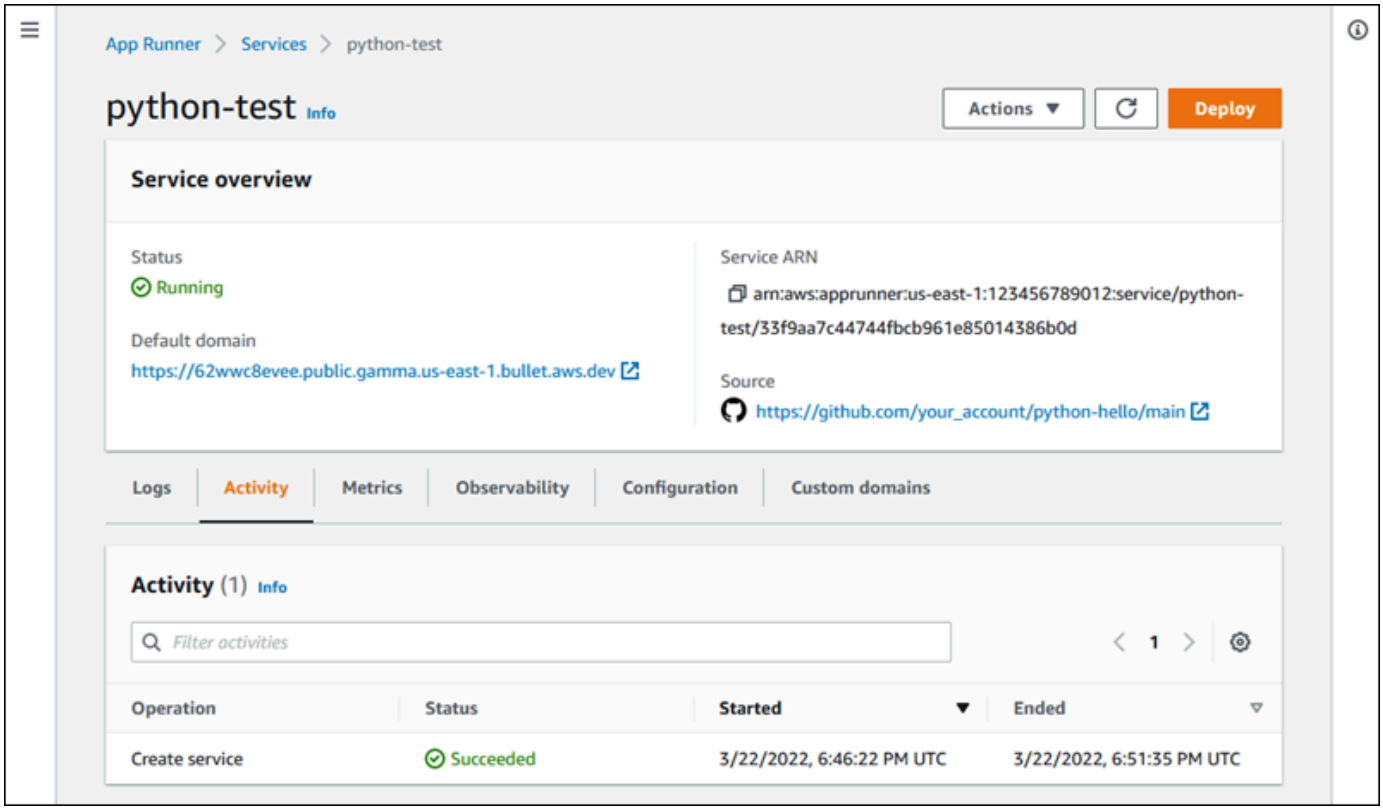
Afficher les journaux d'App Runner dans la console

La console App Runner affiche un résumé de tous les journaux de votre service et vous permet de les consulter, de les explorer et de les télécharger.

Pour consulter les journaux de votre service

1. Ouvrez la console App Runner, puis dans la liste des régions, sélectionnez votre Région AWS.
2. Dans le volet de navigation, choisissez Services, puis choisissez votre service App Runner.

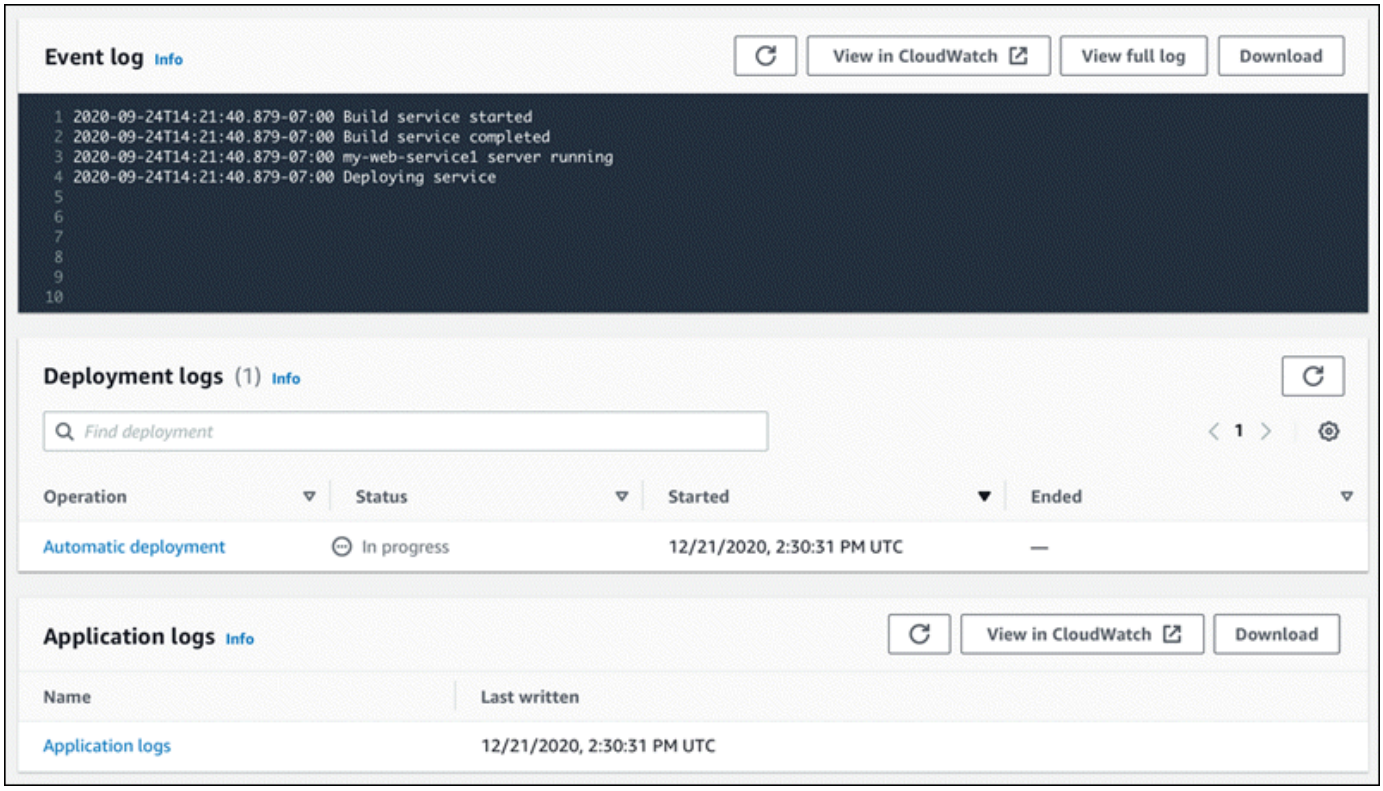
   La console affiche le tableau de bord des services avec une vue d'ensemble des services.



3. Sur la page du tableau de bord du service, choisissez l'onglet Logs.

   La console affiche quelques types de journaux dans plusieurs sections :

   - Journal des événements : activité pendant le cycle de vie de votre service App Runner. La console affiche les derniers événements.
   - Journaux de déploiement : envoyez les déploiements de référentiels à votre service App Runner. La console affiche un flux de journal distinct pour chaque déploiement.
   - Journaux d'application : sortie de l'application Web déployée sur votre service App Runner. La console combine les résultats de toutes les instances en cours d'exécution dans un seul flux de journal.



4. Pour rechercher des déploiements spécifiques, réduisez la liste des journaux de déploiement en saisissant un terme de recherche. Vous pouvez rechercher n'importe quelle valeur figurant dans le tableau.
5. Pour afficher le contenu d'un journal, choisissez Afficher le journal complet (journal des événements) ou le nom du flux du journal (journaux de déploiement et d'application).
6. Choisissez Télécharger pour télécharger un journal. Pour un flux de journal de déploiement, sélectionnez d'abord un flux de journal.
7. Choisissez Afficher dans CloudWatch pour ouvrir la CloudWatch console et utiliser toutes ses fonctionnalités pour explorer les journaux de service de votre App Runner. Pour un flux de journal de déploiement, sélectionnez d'abord un flux de journal.

> **Note**
>
> La CloudWatch console est particulièrement utile si vous souhaitez consulter les journaux d'applications d'instances spécifiques au lieu du journal d'applications combiné.

Affichage des statistiques de service App Runner signalées à CloudWatch

Amazon CloudWatch surveille vos ressources Amazon Web Services (AWS) et les applications que vous utilisez AWS en temps réel. Vous pouvez les utiliser CloudWatch pour collecter et suivre les métriques, qui sont des variables que vous pouvez mesurer pour vos ressources et vos applications. Vous pouvez également l'utiliser pour créer des alarmes qui surveillent les métriques. Lorsqu'un certain seuil est atteint, CloudWatch envoie des notifications ou modifie automatiquement les ressources surveillées. Pour plus d'informations, consultez le guide de CloudWatch l'utilisateur Amazon.

AWS App Runner collecte diverses mesures qui vous offrent une meilleure visibilité sur l'utilisation, les performances et la disponibilité de vos services App Runner. Certains indicateurs suivent les instances individuelles qui exécutent votre service Web, tandis que d'autres concernent le niveau de service global. Les sections suivantes répertorient les statistiques d'App Runner et vous montrent comment les consulter dans la console App Runner.

Statistiques d'App Runner

App Runner collecte les métriques suivantes relatives à votre service et les publie CloudWatch dans l'espace de `AWS/AppRunner` noms.

> **Note**
>
> Avant le 23 août 2023, les mesures d'utilisation du processeur et d'utilisation de la mémoire étaient basées sur les unités de vCPU et les mégaoctets de mémoire utilisés, plutôt que sur le pourcentage d'utilisation tel que calculé aujourd'hui. Si votre application s'est exécutée sur App Runner avant cette date et que vous choisissez de revenir pour consulter les statistiques de cette date sur App Runner ou sur la CloudWatch console, vous verrez un affichage des statistiques dans les deux unités et constaterez également certaines irrégularités en conséquence.

> **Important**
>
> Vous devrez mettre à jour toutes les CloudWatch alarmes basées sur les valeurs métriques d'utilisation du processeur et d'utilisation de la mémoire avant le 23 août 2023. Mettez à jour

les alarmes à déclencher en fonction du pourcentage d'utilisation plutôt que du vCPU ou des mégaoctets. Pour plus d'informations, consultez le guide de CloudWatch l'utilisateur Amazon.

Les métriques au niveau de l'instance sont collectées pour chaque instance (unité de mise à l'échelle) individuellement.

Qu'est-ce qui est mesuré ?	Métrique	Description
CPU utilization	`CPUUtilization`	Pourcentage de l'utilisation moyenne du processeur pendant des périodes d'une minute par rapport à l'utilisation totale du processeur réservée par la configuration du service.
Memory utilization	`MemoryUtilization`	Pourcentage d'utilisation moyenne de la mémoire pendant des périodes d'une minute par rapport à la mémoire totale réservée par la configuration du service.

Les indicateurs de niveau de service sont collectés pour l'ensemble du service.

Qu'est-ce qui est mesuré ?	Métrique	Description
CPU utilization	`CPUUtilization`	Pourcentage d'utilisation agrégée du processeur sur toutes les instances pendant des périodes d'une minute par rapport à l'utilisation totale du processeur réservée par la configuration du service.
Memory utilization	`MemoryUtilization`	Pourcentage d'utilisation de la mémoire agrégée sur toutes les instances pendant des périodes d'une minute par rapport à la mémoire totale réservée par la configuration du service.

Qu'est-ce qui est mesuré ?	Métrique	Description
Concurrency	`Concurrency`	Nombre approximatif de demandes simultanées traitées par le service.
HTTP request count	`Requests`	Le nombre de requêtes HTTP reçues par le service.

HTTP status counts	`2xxStatus Responses`
	
	`4xxStatus Responses`
	
	`5xxStatus Responses`
		Nombre de requêtes HTTP ayant renvoyé l'état de chaque réponse, regroupées par catégorie (2XX, 4XX, 5XX).

HTTP request latency	`RequestLatency`	Le temps, en millisecondes, nécessaire à votre service Web pour traiter les requêtes HTTP.
Instance counts	`ActiveInstances`	Le nombre d'instances qui traitent les requêtes HTTP pour votre service. **Note** Si la `ActiveInstances` métrique affiche zéro, cela signifie qu'il n'y a aucune demande pour le service. Cela n'indique pas que le nombre d'instances de votre service est nul.

Afficher les statistiques d'App Runner dans la console

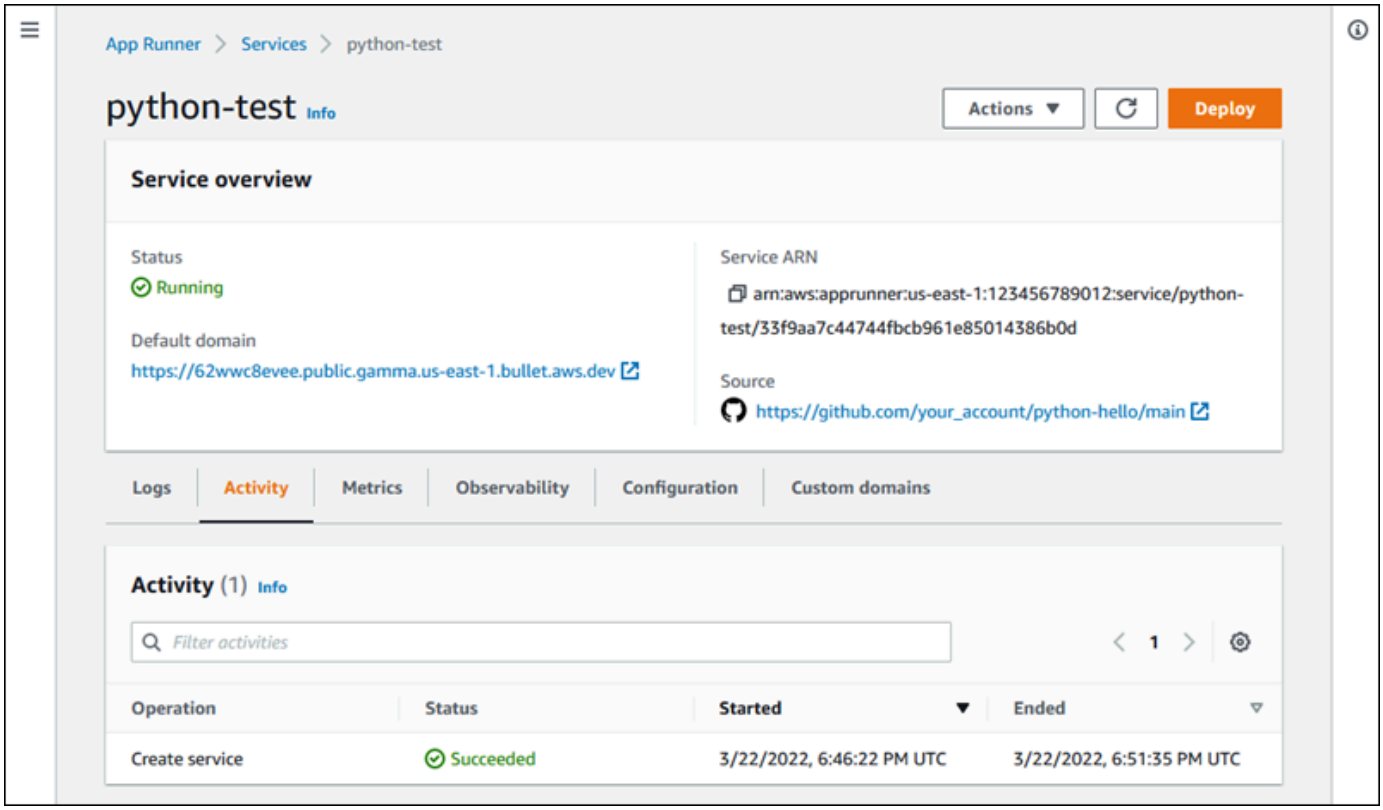
La console App Runner affiche graphiquement les statistiques collectées par App Runner pour votre service et propose d'autres moyens de les explorer.

> **Note**
>
> Pour le moment, la console affiche uniquement les métriques de service. Pour consulter les métriques de l'instance, utilisez la CloudWatch console.

Pour consulter les journaux de votre service

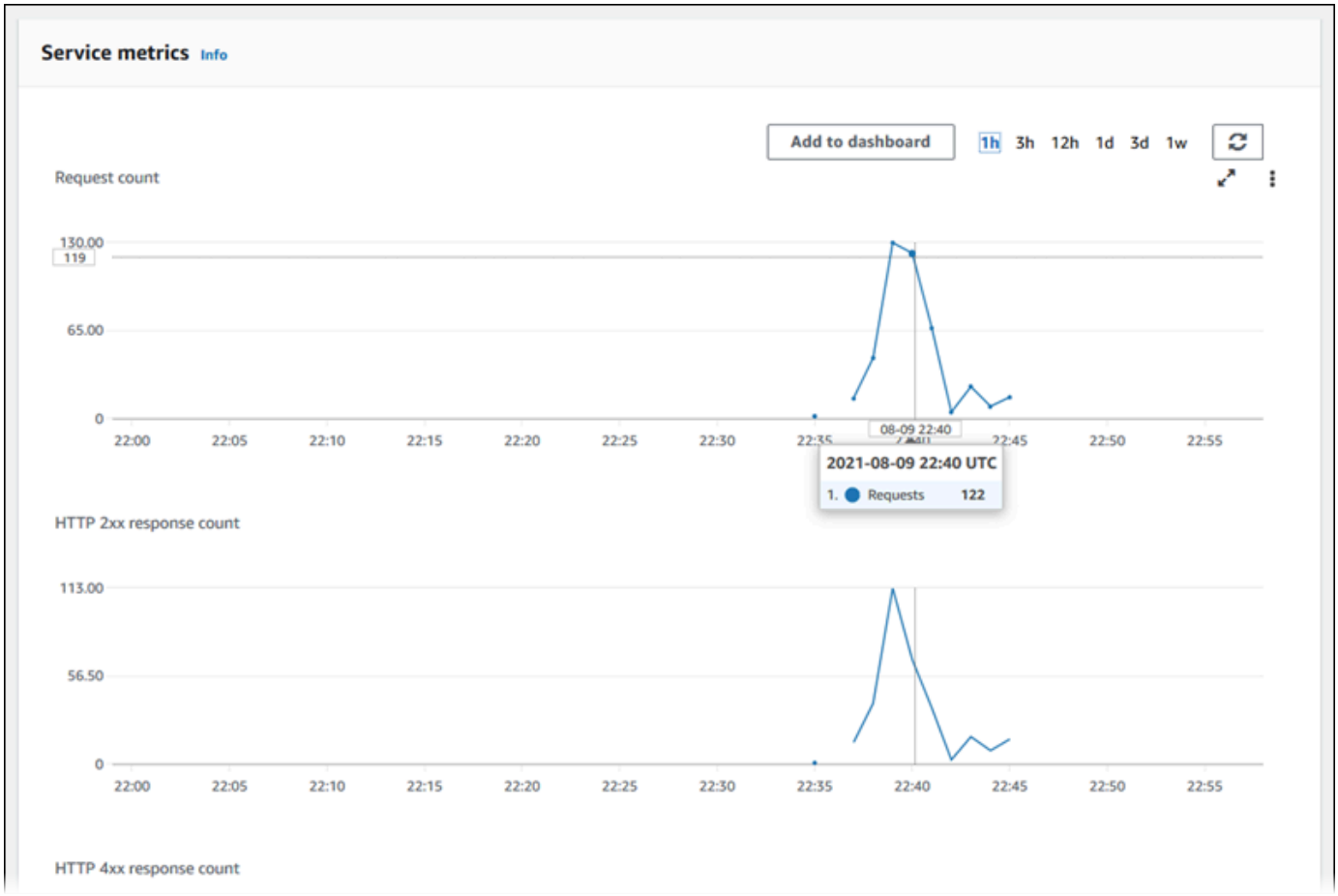
1. Ouvrez la console App Runner, puis dans la liste des régions, sélectionnez votre Région AWS.
2. Dans le volet de navigation, choisissez Services, puis choisissez votre service App Runner.

   La console affiche le tableau de bord des services avec une vue d'ensemble des services.



3. Sur la page du tableau de bord du service, choisissez l'onglet Metrics.

   La console affiche un ensemble de graphiques de mesures.



4. Choisissez une durée (par exemple, 12 h) pour étendre les graphiques de mesures à la période récente de cette durée.
5. Choisissez Ajouter au tableau de bord en haut de l'une des sections du graphique, ou utilisez le menu de n'importe quel graphique pour ajouter les mesures pertinentes à un tableau de bord dans la CloudWatch console afin de poursuivre les recherches.

Gestion des événements App Runner dans EventBridge

À l'aide d'Amazon EventBridge, vous pouvez configurer des règles basées sur les événements qui surveillent un flux de données en temps réel provenant de votre AWS App Runner service afin de détecter certains modèles. Lorsqu'un modèle correspond à une règle, EventBridge lance une action dans une cible telle qu' AWS Lambda Amazon ECS et Amazon SNS. AWS Batch Par exemple, vous pouvez définir une règle pour l'envoi de notifications par e-mail en signalant un sujet Amazon SNS chaque fois qu'un déploiement de votre service échoue. Vous pouvez également définir une fonction Lambda pour avertir une chaîne Slack en cas d'échec d'une mise à jour du service. Pour plus d'informations EventBridge, consultez le guide de EventBridge l'utilisateur Amazon.

App Runner envoie les types d'événements suivants à EventBridge

- Modification de l'état du service : modification du statut d'un service App Runner. Par exemple, l'état d'un service est devenu`DELETE_FAILED`.
- Modification de l'état du fonctionnement du service : modification de l'état d'une longue opération asynchrone sur un service App Runner. Par exemple, un service a commencé à être créé, une mise à jour de service s'est terminée avec succès ou un déploiement de service s'est terminé avec des erreurs.

Création d'une EventBridge règle pour agir sur les événements App Runner

Un EventBridge événement est un objet qui définit certains EventBridge champs standard, tels que le AWS service source et le type de détail (événement), ainsi qu'un ensemble de champs spécifiques à l'événement contenant les détails de l'événement. Pour créer une EventBridge règle, vous utilisez la EventBridge console pour définir un modèle d'événements (quels événements doivent être suivis) et spécifier une action cible (ce qui doit être fait lors d'un match). Un modèle d'événement est similaire aux événements auxquels il correspond. Vous spécifiez un sous-ensemble de champs à associer, et pour chaque champ, vous spécifiez une liste de valeurs possibles. Cette rubrique fournit des exemples d'événements et de modèles d'événements App Runner.

Pour plus d'informations sur la création de EventBridge règles, consultez la section Création d'une règle pour un AWS service dans le guide de EventBridge l'utilisateur Amazon.

> **Note**
>
> Certains services prennent en charge des modèles prédéfinis dans. EventBridge Cela simplifie la création d'un modèle d'événement. Vous sélectionnez les valeurs des champs dans un formulaire et vous EventBridge générez le modèle pour vous. Pour le moment, App Runner ne prend pas en charge les modèles prédéfinis. Vous devez saisir le modèle sous forme d'objet JSON. Vous pouvez utiliser les exemples présentés dans cette rubrique comme point de départ.

Exemples d'événements App Runner

Voici quelques exemples d'événements auxquels App Runner envoie des messages EventBridge.

- Un événement de modification de l'état du service. Plus précisément, un service qui est passé du `RUNNING` statut `OPERATION_IN_PROGRESS` au statut.

```
{
  "version": "0",
  "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
  "detail-type": "AppRunner Service Status Change",
  "source": "aws.apprunner",
  "account": "111122223333",
  "time": "2021-04-29T11:54:23Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:apprunner:us-east-2:123456789012:service/my-
app/8fe1e10304f84fd2b0df550fe98a71fa"
  ],
  "detail": {
    "previousServiceStatus": "OPERATION_IN_PROGRESS",
    "currentServiceStatus": "RUNNING",
    "serviceName": "my-app",
    "serviceId": "8fe1e10304f84fd2b0df550fe98a71fa",
    "message": "Service status is set to RUNNING.",
    "severity": "INFO"
  }
}
```

- Un événement de modification du statut de l'opération. Plus précisément, une `UpdateService` opération qui s'est terminée avec succès.

```
{
  "version": "0",
  "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
  "detail-type": "AppRunner Service Operation Status Change",
  "source": "aws.apprunner",
  "account": "111122223333",
  "time": "2021-04-29T18:43:48Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:apprunner:us-east-2:123456789012:service/my-
app/8fe1e10304f84fd2b0df550fe98a71fa"
  ],
  "detail": {
    "operationStatus": "UpdateServiceCompletedSuccessfully",
    "serviceName": "my-app",
```

```
    "serviceId": "8fe1e10304f84fd2b0df550fe98a71fa",
    "message": "Service update completed successfully. New application and
 configuration is deployed.",
    "severity": "INFO"
  }
}
```

Exemples de modèles d'événements App Runner

Les exemples suivants illustrent les modèles d'événements que vous pouvez utiliser dans EventBridge les règles pour correspondre à un ou plusieurs événements App Runner. Un modèle d'événement est similaire à un événement. N'incluez que les champs auxquels vous souhaitez faire correspondre et fournissez une liste au lieu d'un scalaire pour chacun d'entre eux.

- Faites correspondre tous les événements de changement de statut de service pour les services d'un compte spécifique, lorsque le service n'est plus en `RUNNING` statut.

```
{
  "detail-type": [ "AppRunner Service Status Change" ],
  "source": [ "aws.apprunner" ],
  "account": [ "111122223333" ],
  "detail": {
    "previousServiceStatus": [ "RUNNING" ]
  }
}
```

- Faites correspondre tous les événements de changement de statut d'opération pour les services d'un compte spécifique, lorsque l'opération a échoué.

```
{
  "detail-type": [ "AppRunner Service Operation Status Change" ],
  "source": [ "aws.apprunner" ],
  "account": [ "111122223333" ],
  "detail": {
    "operationStatus": [
      "CreateServiceFailed",
      "DeleteServiceFailed",
      "UpdateServiceFailed",
      "DeploymentFailed",
      "PauseServiceFailed",
      "ResumeServiceFailed"
```

```
    ]
  }
}
```

Référence des événements App Runner

Modification de l'état du service

Un événement de modification de l'état du service est `detail-type` défini sur`AppRunner Service Status Change`. Il comporte les champs de détail et les valeurs suivants :

```
"serviceId": "your service ID",
"serviceName": "your service name",
"message": "Service status is set to CurrentStatus.",
"previousServiceStatus": "any valid service status",
"currentServiceStatus": "any valid service status",
"severity": "varies"
```

Modification du statut de l'opération

Un événement de modification du statut de l'opération est `detail-type` défini sur`AppRunner Service Operation Status Change`. Il comporte les champs de détail et les valeurs suivants :

```
"operationStatus": "see following table",
"serviceName": "your service name",
"serviceId": "your service ID",
"message": "see following table",
"severity": "varies"
```

Le tableau suivant répertorie tous les codes d'état possibles et les messages associés.

Statut	Message
`CreateServiceStarted`	La création du service a commencé.
`CreateServiceCompletedSuccessfully`	La création du service s'est terminée avec succès.

Statut	Message
`CreateServiceFailed`	La création du service a échoué. Pour plus de détails, consultez les journaux de service.
`DeleteServiceStarted`	La suppression du service a commencé.
`DeleteServiceCompletedSuccessfully`	La suppression du service s'est terminée avec succès.
`DeleteServiceFailed`	La suppression du service a échoué.
`UpdateServiceStarted`	
`UpdateServiceCompletedSuccessfully`	La mise à jour du service s'est terminée correctement. Une nouvelle application et une nouvelle configuration sont déployées.
	La mise à jour du service s'est terminée correctement. La nouvelle configuration est déployée.
`UpdateServiceFailed`	La mise à jour du service a échoué. Pour plus de détails, consultez les journaux de service.
`DeploymentStarted`	Le déploiement a commencé.
`DeploymentCompletedSuccessfully`	Le déploiement s'est terminé avec succès.
`DeploymentFailed`	Le déploiement a échoué. Pour plus de détails, consultez les journaux de service.
`PauseServiceStarted`	La pause de service a commencé.
`PauseServiceCompletedSuccessfully`	La pause du service s'est terminée avec succès.
`PauseServiceFailed`	La pause du service a échoué.
`ResumeServiceStarted`	La reprise du service a commencé.

Statut	Message
`ResumeServiceCompletedSuccessfully`	Reprise du service terminée avec succès.
`ResumeServiceFailed`	La reprise du service a échoué.

Journalisation des appels d'API App Runner avec AWS CloudTrail

App Runner est intégré à AWS CloudTrail un service qui fournit un enregistrement des actions effectuées par un utilisateur, un rôle ou un AWS service dans App Runner. CloudTrailcapture tous les appels d'API pour App Runner sous forme d'événements. Les appels capturés incluent des appels provenant de la console App Runner et des appels de code vers les opérations de l'API App Runner. Si vous créez un suivi, vous pouvez activer la diffusion continue d' CloudTrail événements vers un compartiment Amazon S3, y compris des événements pour App Runner. Si vous ne configurez pas de suivi, vous pouvez toujours consulter les événements les plus récents dans la CloudTrail console dans Historique des événements. À l'aide des informations collectées par CloudTrail, vous pouvez déterminer la demande qui a été faite à App Runner, l'adresse IP à partir de laquelle la demande a été faite, qui a fait la demande, quand elle a été faite et des détails supplémentaires.

Pour en savoir plus CloudTrail, consultez le guide de AWS CloudTrail l'utilisateur.

Informations sur App Runner dans CloudTrail

CloudTrail est activé sur votre compte Compte AWS lorsque vous créez le compte. Lorsqu'une activité se produit dans App Runner, cette activité est enregistrée dans un CloudTrail événement avec d'autres événements de AWS service dans l'historique des événements. Vous pouvez consulter, rechercher et télécharger les événements récents dans votre Compte AWS. Pour plus d'informations, consultez la section Affichage des événements avec l'historique des CloudTrail événements.

Pour un enregistrement continu des événements de votre navigateur Compte AWS, y compris des événements liés à App Runner, créez un parcours. Un suivi permet CloudTrail de fournir des fichiers journaux à un compartiment Amazon S3. Par défaut, lorsque vous créez un journal d’activité dans la console, il s’applique à toutes les régions Régions AWS. Le journal enregistre les événements de toutes les régions de la AWS partition et transmet les fichiers journaux au compartiment Amazon S3 que vous spécifiez. En outre, vous pouvez configurer d'autres AWS services pour analyser plus en

détail les données d'événements collectées dans les CloudTrail journaux et agir en conséquence. Pour plus d'informations, consultez les ressources suivantes :

- Vue d'ensemble de la création d'un journal d'activité
- CloudTrail Services et intégrations pris en charge
- Configuration des notifications Amazon SNS pour CloudTrail
- Réception de fichiers CloudTrail journaux de plusieurs régions et réception de fichiers CloudTrail journaux de plusieurs comptes

Toutes les actions d'App Runner sont enregistrées CloudTrail et documentées dans la référence de l' AWS App Runner API. Par exemple, les appels aux `CreateServiceDeleteConnection`, et `StartDeployment` les actions génèrent des entrées dans les fichiers CloudTrail journaux.

Chaque événement ou entrée de journal contient des informations sur la personne ayant initié la demande. Les informations relatives à l'identité permettent de déterminer les éléments suivants :

- Si la demande a été effectuée avec des informations d'identification d'utilisateur root ou IAM.
- Si la demande a été effectuée avec des informations d'identification de sécurité temporaires pour un rôle ou un utilisateur fédéré.
- Si la demande a été faite par un autre AWS service.

Pour plus d'informations, consultez la section Élément userIdentity CloudTrail .

Comprendre les entrées du fichier journal d'App Runner

Un suivi est une configuration qui permet de transmettre des événements sous forme de fichiers journaux à un compartiment Amazon S3 que vous spécifiez. CloudTrail les fichiers journaux contiennent une ou plusieurs entrées de journal. Un événement représente une demande unique provenant de n'importe quelle source et inclut des informations sur l'action demandée, la date et l'heure de l'action, ainsi que les paramètres de la demande. CloudTrail les fichiers journaux ne constituent pas une trace ordonnée des appels d'API publics, ils n'apparaissent donc pas dans un ordre spécifique.

L'exemple suivant montre une entrée de CloudTrail journal illustrant l'`CreateService`action.

> **Note**
>
> Pour des raisons de sécurité, certaines valeurs de propriétés sont supprimées dans les journaux et remplacées par le texte`HIDDEN_DUE_TO_SECURITY_REASONS`. Cela empêche la divulgation involontaire d'informations secrètes. Cependant, vous pouvez toujours voir que ces propriétés ont été transmises dans la demande ou renvoyées dans la réponse.

Exemple d'entrée de CloudTrail journal pour l'action **`CreateService`** App Runner

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:iam::123456789012:user/aws-user",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "aws-user"
  },
  "eventTime": "2020-10-02T23:25:33Z",
  "eventSource": "apprunner.amazonaws.com",
  "eventName": "CreateService",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_6) AppleWebKit/537.36
 (KHTML, like Gecko) Chrome/77.0.3865.75 Safari/537.36",
  "requestParameters": {
    "serviceName": "python-test",
    "sourceConfiguration": {
      "codeRepository": {
        "repositoryUrl": "https://github.com/github-user/python-hello",
        "sourceCodeVersion": {
          "type": "BRANCH",
          "value": "main"
        },
        "codeConfiguration": {
          "configurationSource": "API",
          "codeConfigurationValues": {
            "runtime": "python3",
            "buildCommand": "HIDDEN_DUE_TO_SECURITY_REASONS",
            "startCommand": "HIDDEN_DUE_TO_SECURITY_REASONS",
```

```
            "port": "8080",
            "runtimeEnvironmentVariables": "HIDDEN_DUE_TO_SECURITY_REASONS"
          }
        }
      },
      "autoDeploymentsEnabled": true,
      "authenticationConfiguration": {
        "connectionArn": "arn:aws:apprunner:us-east-2:123456789012:connection/your-
connection/e7656250f67242d7819feade6800f59e"
      }
    },
    "healthCheckConfiguration": {
      "protocol": "HTTP"
    },
    "instanceConfiguration": {
      "cpu": "256",
      "memory": "1024"
    }
  },
  "responseElements": {
    "service": {
        "serviceName": "python-test",
        "serviceId": "dfa2b7cc7bcb4b6fa6c1f0f4efff988a",
        "serviceArn": "arn:aws:apprunner:us-east-2:123456789012:service/python-test/
dfa2b7cc7bcb4b6fa6c1f0f4efff988a",
        "serviceUrl": "generated domain",
        "createdAt": "2020-10-02T23:25:32.650Z",
        "updatedAt": "2020-10-02T23:25:32.650Z",
        "status": "OPERATION_IN_PROGRESS",
        "sourceConfiguration": {
            "codeRepository": {
                "repositoryUrl": "https://github.com/github-user/python-hello",
                "sourceCodeVersion": {
                    "type": "Branch",
                    "value": "main"
                },
                "sourceDirectory": "/",
                "codeConfiguration": {
                    "codeConfigurationValues": {
                        "configurationSource": "API",
                        "runtime": "python3",
                        "buildCommand": "HIDDEN_DUE_TO_SECURITY_REASONS",
                        "startCommand": "HIDDEN_DUE_TO_SECURITY_REASONS",
                        "port": "8080",
```

```
                        "runtimeEnvironmentVariables": "HIDDEN_DUE_TO_SECURITY_REASONS"
                    }
                }
            },
            "autoDeploymentsEnabled": true,
            "authenticationConfiguration": {
                "connectionArn": "arn:aws:apprunner:us-east-2:123456789012:connection/
your-connection/e7656250f67242d7819feade6800f59e"
            }
        },
        "healthCheckConfiguration": {
            "protocol": "HTTP",
            "path": "/",
            "interval": 5,
            "timeout": 2,
            "healthyThreshold": 3,
            "unhealthyThreshold": 5
        },
        "instanceConfiguration": {
            "cpu": "256",
            "memory": "1024"
        },
        "autoScalingConfigurationSummary": {
            "autoScalingConfigurationArn": "arn:aws:apprunner:us-
east-2:123456789012:autoscalingconfiguration/
DefaultConfiguration/1/00000000000000000000000000000001",
            "autoScalingConfigurationName": "DefaultConfiguration",
            "autoScalingConfigurationRevision": 1
        }
    }
},
  "requestID": "1a60af60-ecf5-4280-aa8f-64538319ba0a",
  "eventID": "e1a3f623-4d24-4390-a70b-bf08a0e24669",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```

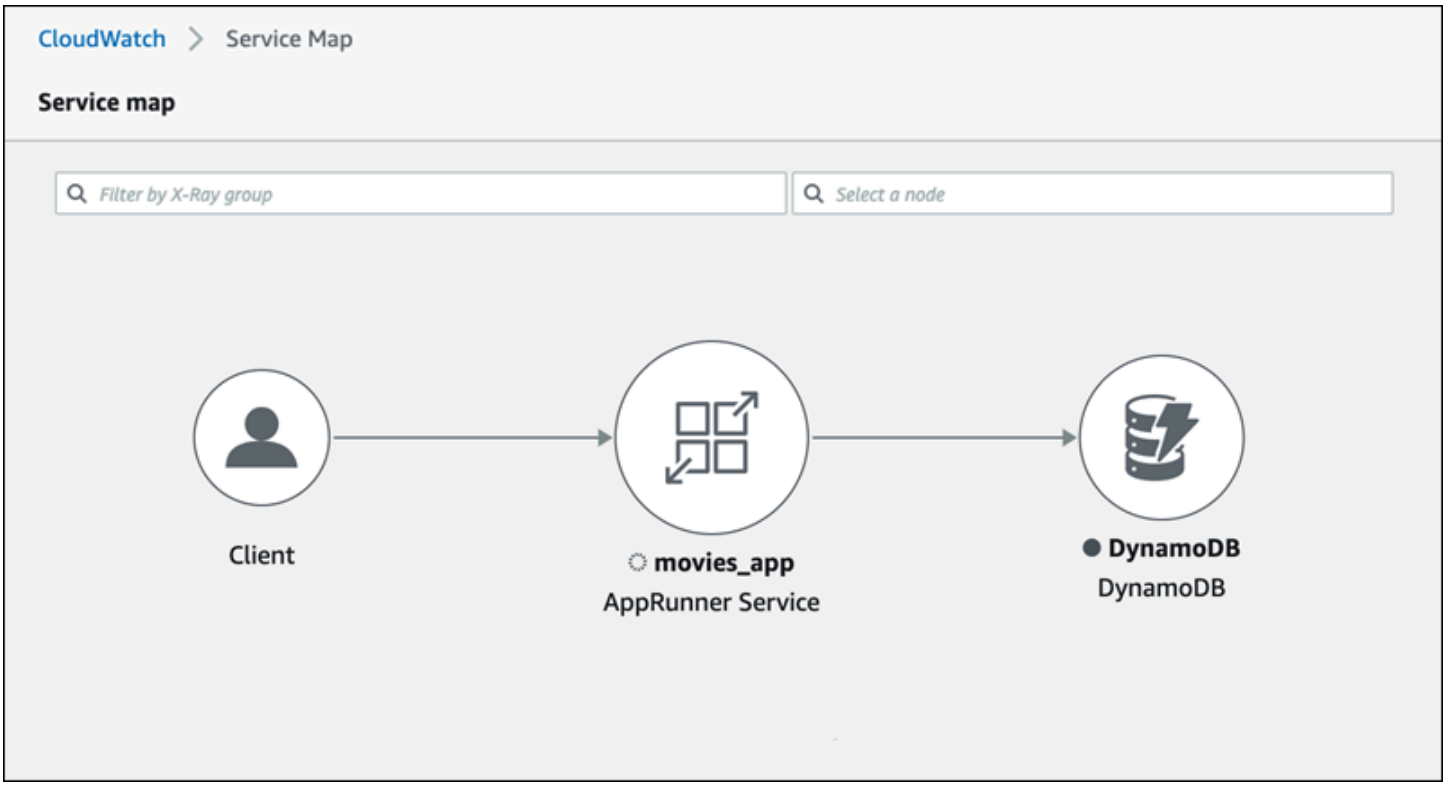
Suivi de votre application App Runner avec X-Ray

AWS X-Ray est un service qui collecte des données sur les demandes traitées par votre application et fournit des outils que vous pouvez utiliser pour visualiser, filtrer et obtenir des informations sur

ces données afin d'identifier les problèmes et les opportunités d'optimisation. Pour toute demande retracée envoyée à votre application, vous pouvez consulter des informations détaillées non seulement sur la demande et la réponse, mais également sur les appels que votre application fait aux AWS ressources en aval, aux microservices, aux bases de données et au Web APIs HTTP.

X-Ray utilise les données de suivi provenant des AWS ressources qui alimentent vos applications cloud pour générer un graphique de service détaillé. Le graphique de services montre le client, le service frontal et les services dorsaux que votre service frontal appelle pour traiter les demandes et conserver les données. Utilisez le graphique de services pour identifier les goulots d'étranglement, les pics de latence et d'autres problèmes à résoudre pour améliorer les performances de vos applications.

Pour plus d'informations sur X-Ray, veuillez consulter le Guide du développeur AWS X-Ray.



Instrumentez votre application pour le traçage

Instrumentez votre application de service App Runner pour le traçage à l'aide OpenTelemetryd'une spécification de télémétrie portable. À l'heure actuelle, App Runner prend en charge AWS Distro for OpenTelemetry (ADOT), une OpenTelemetry implémentation qui collecte et présente des informations de télémétrie à l'aide de services. AWS X-Ray implémente le composant de traçage.

Selon le SDK ADOT spécifique que vous utilisez dans votre application, ADOT prend en charge jusqu'à deux approches d'instrumentation : automatique et manuelle. Pour plus d'informations sur l'instrumentation avec votre SDK, consultez la documentation ADOT et choisissez votre SDK dans le volet de navigation.

Configuration du moteur d'exécution

Vous trouverez ci-dessous les instructions générales de configuration de l'exécution pour instrumenter votre application de service App Runner à des fins de traçage.

Pour configurer le suivi pour votre environnement d'exécution

1. Suivez les instructions fournies pour votre environnement d'exécution dans AWS Distro for OpenTelemetry (ADOT) pour instrumenter votre application.
2. Installez les `OTEL` dépendances requises dans la `build` section du `apprunner.yaml` fichier si vous utilisez le référentiel de code source ou dans le Dockerfile si vous utilisez une image de conteneur.
3. Configurez vos variables d'environnement dans le `apprunner.yaml` fichier si vous utilisez le référentiel de code source ou dans le Dockerfile si vous utilisez une image de conteneur.

   Example Variables d'environnement

   > **Note**
   >
   > L'exemple suivant répertorie les variables d'environnement importantes à ajouter au `apprunner.yaml` fichier. Ajoutez ces variables d'environnement à votre Dockerfile si vous utilisez une image de conteneur. Cependant, chaque environnement d'exécution peut avoir ses propres particularités et vous devrez peut-être ajouter d'autres variables d'environnement à la liste suivante. Pour plus d'informations sur les instructions spécifiques à votre environnement d'exécution et des exemples sur la façon de configurer votre application pour votre environnement d'exécution, consultez AWS Distro for OpenTelemetry et accédez à votre environnement d'exécution, sous Getting Started.

```
env:
    - name: OTEL_PROPAGATORS
      value: xray
    - name: OTEL_METRICS_EXPORTER
      value: none
```

```
    - name: OTEL_EXPORTER_OTLP_ENDPOINT
      value: http://localhost:4317
    - name: OTEL_RESOURCE_ATTRIBUTES
      value: 'service.name=example_app'
```

> **Note**
>
> `OTEL_METRICS_EXPORTER=none`est une variable d'environnement importante pour App Runner, car le collecteur App Runner Otel n'accepte pas l'enregistrement des métriques. Il n'accepte que le suivi des métriques.

Exemple de configuration d'exécution

L'exemple suivant illustre l'instrumentation automatique de votre application avec le SDK ADOT Python. Le SDK produit automatiquement des intervalles avec des données de télémétrie décrivant les valeurs utilisées par les frameworks Python dans votre application sans ajouter une seule ligne de code Python. Il suffit d'ajouter ou de modifier quelques lignes dans deux fichiers sources.

Ajoutez d'abord quelques dépendances, comme indiqué dans l'exemple suivant.

Example requirements.txt

```
opentelemetry-distro[otlp]>=0.24b0
opentelemetry-sdk-extension-aws~=2.0
opentelemetry-propagator-aws-xray~=1.0
```

Ensuite, instrumentez votre application. La manière de procéder dépend de la source de votre service : image source ou code source.

Source image

Lorsque la source de votre service est une image, vous pouvez directement instrumenter le Dockerfile qui contrôle la création de votre image de conteneur et l'exécution de l'application dans l'image. L'exemple suivant montre un Dockerfile instrumenté pour une application Python. Les ajouts à l'instrumentation sont soulignés en gras.

Example Dockerfile

```
FROM public.ecr.aws/amazonlinux/amazonlinux:latest
```

```
RUN yum install python3.7 -y && curl -O https://bootstrap.pypa.io/get-pip.py &&
 python3 get-pip.py && yum update -y
COPY . /app
WORKDIR /app
RUN pip3 install -r requirements.txt
RUN opentelemetry-bootstrap --action=install
ENV OTEL_PYTHON_DISABLED_INSTRUMENTATIONS=urllib3
ENV OTEL_METRICS_EXPORTER=none
ENV OTEL_RESOURCE_ATTRIBUTES='service.name=example_app'
CMD OTEL_PROPAGATORS=xray OTEL_PYTHON_ID_GENERATOR=xray opentelemetry-instrument
 python3 app.py
EXPOSE 8080
```

Source code repository

Lorsque votre source de service est un référentiel contenant la source de votre application, vous instrumentez indirectement votre image à l'aide des paramètres du fichier de configuration d'App Runner. Ces paramètres contrôlent le Dockerfile qu'App Runner génère et utilise pour créer l'image de votre application. L'exemple suivant montre un fichier de configuration App Runner instrumenté pour une application Python. Les ajouts à l'instrumentation sont soulignés en gras.

Example apprunner.yaml

```
version: 1.0
runtime: python3
build:
  commands:
    build:
      - pip install -r requirements.txt
      - opentelemetry-bootstrap --action=install
run:
  command: opentelemetry-instrument python app.py
  network:
    port: 8080
  env:
    - name: OTEL_PROPAGATORS
      value: xray
    - name: OTEL_METRICS_EXPORTER
      value: none
    - name: OTEL_PYTHON_ID_GENERATOR
      value: xray
    - name: OTEL_PYTHON_DISABLED_INSTRUMENTATIONS
```

```
    value: urllib3
  - name: OTEL_RESOURCE_ATTRIBUTES
    value: 'service.name=example_app'
```

Ajoutez des autorisations X-Ray à votre rôle d'instance de service App Runner

Pour utiliser le traçage X-Ray avec votre service App Runner, vous devez fournir aux instances du service des autorisations leur permettant d'interagir avec le service X-Ray. Pour ce faire, associez un rôle d'instance à votre service et ajoutez une politique gérée avec des autorisations X-Ray. Pour plus d'informations sur un rôle d'instance App Runner, consultezthe section called “Rôle d’instance”. Ajoutez la politique `AWSXRayDaemonWriteAccess` gérée à votre rôle d'instance et attribuez-la à votre service lors de sa création.

Activez le suivi X-Ray pour votre service App Runner

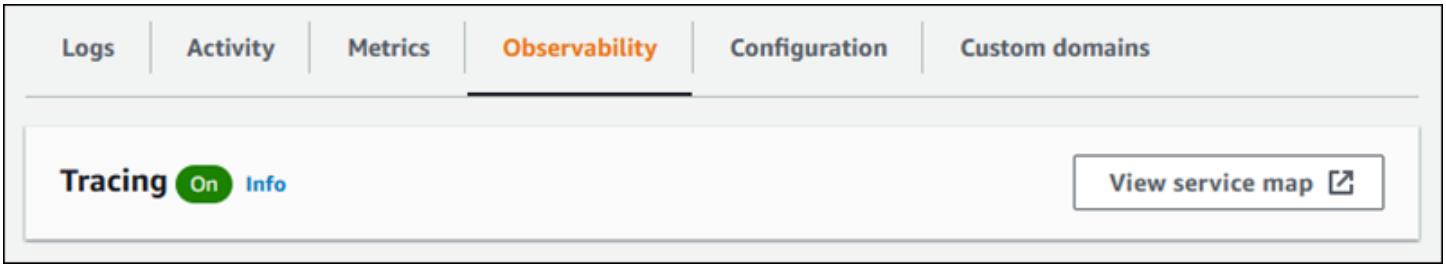
Lorsque vous créez un service, App Runner désactive le suivi par défaut. Vous pouvez activer le suivi X-Ray pour votre service dans le cadre de la configuration de l'observabilité. Pour de plus amples informations, veuillez consulter the section called “Gérez l'observabilité”.

Si vous utilisez l'API App Runner ou le AWS CLI, l'TraceConfigurationobjet contenu dans l'objet de ObservabilityConfigurationressource contient des paramètres de suivi. Pour que le traçage reste désactivé, ne spécifiez aucun `TraceConfiguration` objet.

Dans le cas de la console et de l'API, veillez à associer le rôle d'instance décrit dans la section précédente à votre service App Runner.

Afficher les données de suivi X-Ray pour votre service App Runner

Dans l'onglet Observabilité de la page du tableau de bord des services de la console App Runner, choisissez Afficher la carte des services pour accéder à la CloudWatch console Amazon.



Utilisez la CloudWatch console Amazon pour consulter les cartes des services et les traces des demandes traitées par votre application. Les cartes des services présentent des informations telles que la latence des demandes et les interactions avec d'autres applications et AWS services. Les annotations personnalisées que vous ajoutez à votre code vous permettent de rechercher facilement des traces. Pour plus d'informations, consultez la section Utilisation ServiceLens pour surveiller l'état de santé de vos applications dans le guide de CloudWatch l'utilisateur Amazon.

Associer une ACL AWS WAF Web à votre service

AWS WAF est un pare-feu d'applications Web que vous pouvez utiliser pour sécuriser votre service App Runner. Grâce aux listes de contrôle d'accès AWS WAF Web (Web ACLs), vous pouvez protéger les points de terminaison de votre service App Runner contre les exploits Web courants et les robots indésirables.

Une ACL Web vous permet de contrôler avec précision toutes les requêtes Web entrantes adressées à votre service App Runner. Vous pouvez définir des règles dans une ACL Web pour autoriser, bloquer ou surveiller le trafic Web, afin de garantir que seules les demandes autorisées et légitimes atteignent vos applications Web et APIs. Vous pouvez personnaliser les règles ACL Web en fonction de vos besoins commerciaux et de sécurité spécifiques. Pour en savoir plus sur la sécurité de l'infrastructure et les meilleures pratiques en matière d'application du réseau ACLs, consultez la section Contrôler le trafic réseau dans le guide de l'utilisateur Amazon VPC.

> **Important**
>
> Les règles d'adresse IP source pour les services privés App Runner associés au Web WAF ACLs ne respectent pas les règles basées sur l'adresse IP. Cela est dû au fait que nous ne prenons actuellement pas en charge le transfert des données IP source des demandes vers les services privés App Runner associés à WAF. Si votre application App Runner nécessite des règles de contrôle du trafic IP/CIDR entrant à la source, vous devez utiliser des règles de groupe de sécurité pour les points de terminaison privés plutôt que pour le Web WAF. ACLs

Flux de requêtes Web entrantes

Lorsqu'une ACL AWS WAF Web est associée à un service App Runner, les requêtes Web entrantes suivent le processus suivant :

1. App Runner transmet le contenu de la demande d'origine à AWS WAF.
2. AWS WAF inspecte la demande et compare son contenu aux règles que vous avez spécifiées dans votre ACL Web.
3. Sur la base de son inspection, AWS WAF renvoie une `block` réponse `allow` ou à App Runner.
   - Si une `allow` réponse est renvoyée, App Runner transmet la demande à votre application.

- Si une `block` réponse est renvoyée, App Runner empêche la demande d'atteindre votre application Web. Il transmet la `block` réponse AWS WAF à votre candidature.

> **Note**
>
> Par défaut, App Runner bloque la demande si aucune réponse n'est renvoyée par AWS WAF.

Pour plus d'informations sur le AWS WAF Web ACLs, consultez la section Listes de contrôle d'accès Web (Web ACLs) dans le manuel du AWS WAF développeur.

> **Note**
>
> Vous payez le AWS WAF prix standard. L'utilisation du AWS WAF Web ACLs pour vos services App Runner n'entraîne aucun coût supplémentaire. Pour plus d'informations sur la tarification, consultez la section AWS WAF Tarification.

Associer le Web WAF ACLs à votre service App Runner

Voici le processus de haut niveau permettant d'associer une ACL AWS WAF Web à votre service App Runner :

1. Créez une ACL Web dans la AWS WAF console. Pour plus d'informations, consultez la section Création d'une ACL Web dans le manuel du AWS WAF développeur.
2. Mettez à jour vos autorisations Gestion des identités et des accès AWS (IAM) pour AWS WAF. Pour plus d'informations, consultez Autorisations .
3. Associez l'ACL Web au service App Runner à l'aide de l'une des méthodes suivantes :
   - Console App Runner : associez une ACL Web existante à l'aide de la console App Runner lorsque vous créez ou mettez à jour un service App Runner. Pour obtenir des instructions, consultez la section Gestion AWS WAF du Web ACLs.
   - AWS WAF console : associez l'ACL Web à l'aide de la AWS WAF console d'un service App Runner existant. Pour plus d'informations, consultez la section Associer ou dissocier une ACL Web à une ressource AWS dans le manuel du AWS WAF développeur.

- AWS CLI: associez l'ACL Web à l'aide du AWS WAF public APIs. Pour plus d'informations sur le AWS WAF public APIs, consultez la section AssociateWebACL dans le guide de référence des AWS WAF API.

Considérations

- Les règles d'adresse IP source pour les services privés App Runner associés au Web WAF ACLs ne respectent pas les règles basées sur l'adresse IP. Cela est dû au fait que nous ne prenons actuellement pas en charge le transfert des données IP source des demandes vers les services privés App Runner associés à WAF. Si votre application App Runner nécessite des règles de contrôle du trafic IP/CIDR entrant à la source, vous devez utiliser des règles de groupe de sécurité pour les points de terminaison privés plutôt que pour le Web WAF. ACLs
- Un service App Runner ne peut être associé qu'à une seule ACL Web. Cependant, vous pouvez associer une ACL Web à plusieurs services App Runner et à plusieurs AWS ressources. Les exemples incluent les groupes d'utilisateurs Amazon Cognito et les ressources Application Load Balancer.
- Lorsque vous créez une ACL Web, un court laps de temps s'écoule avant que l'ACL Web ne se propage complètement et ne soit disponible pour App Runner. Le temps de propagation peut aller de quelques secondes à plusieurs minutes. AWS WAF renvoie un `WAFUnavailableEntityException` lorsque vous essayez d'associer une ACL Web avant qu'elle ne soit complètement propagée.

  Si vous actualisez le navigateur ou quittez la console App Runner avant que l'ACL Web ne soit complètement propagée, l'association ne se produit pas. Cependant, vous pouvez naviguer dans la console App Runner.
- AWS WAF renvoie une `WAFNonexistentItemException` erreur lorsque vous appelez l'une des commandes suivantes AWS WAF APIs pour un service App Runner dont l'état n'est pas valide :
  - `AssociateWebACL`
  - `DisassociateWebACL`
  - `GetWebACLForResource`

  Les états non valides pour votre service App Runner sont les suivants :
  - `CREATE_FAILED`
  - `DELETE_FAILED`
  - `DELETED`

- `OPERATION_IN_PROGRESS`

> **Note**
>
> `OPERATION_IN_PROGRESS`l'état n'est pas valide uniquement si votre service App Runner est supprimé.

- Votre demande peut entraîner une charge utile supérieure aux limites de ce qui AWS WAF peut être inspecté. Pour plus d'informations sur le traitement des AWS WAF demandes surdimensionnées provenant d'App Runner, consultez la section Gestion des composants de demandes surdimensionnées dans le Guide du AWS WAF développeur pour savoir comment gérer les demandes AWS WAF surdimensionnées provenant d'App Runner.
- Si vous ne définissez pas les règles appropriées ou si vos modèles de trafic changent, une ACL Web risque de ne pas être aussi efficace pour sécuriser votre application.

Permissions

Pour utiliser une ACL Web dans AWS App Runner, ajoutez les autorisations IAM suivantes pour AWS WAF :

- `apprunner:ListAssociatedServicesForWebAcl`
- `apprunner:DescribeWebAclForService`
- `apprunner:AssociateWebAcl`
- `apprunner:DisassociateWebAcl`

Pour plus d'informations sur les autorisations IAM, consultez la section Politiques et autorisations dans IAM dans le guide de l'utilisateur IAM.

Voici un exemple de la politique IAM mise à jour pour AWS WAF. Cette politique IAM inclut les autorisations nécessaires pour utiliser un service App Runner.

Example

JSON

```
{
```

```
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "wafv2:ListResourcesForWebACL",
            "wafv2:GetWebACLForResource",
            "wafv2:AssociateWebACL",
            "wafv2:DisassociateWebACL",
            "apprunner:ListAssociatedServicesForWebAcl",
            "apprunner:DescribeWebAclForService",
            "apprunner:AssociateWebAcl",
            "apprunner:DisassociateWebAcl"
         ],
         "Resource":"*"
      }
   ]
}
```

> **Note**
>
> Bien que vous deviez accorder des autorisations IAM, les actions répertoriées sont uniquement des autorisations et ne correspondent à aucune opération d'API.

Gérer AWS WAF le Web ACLs

Gérez le AWS WAF Web ACLs pour votre service App Runner en utilisant l'une des méthodes suivantes :

- the section called "Console App Runner"
- the section called "AWS CLI"

Console App Runner

Lorsque vous créez un service ou que vous mettez à jour un service existant sur la console App Runner, vous pouvez associer ou dissocier une ACL AWS WAF Web.

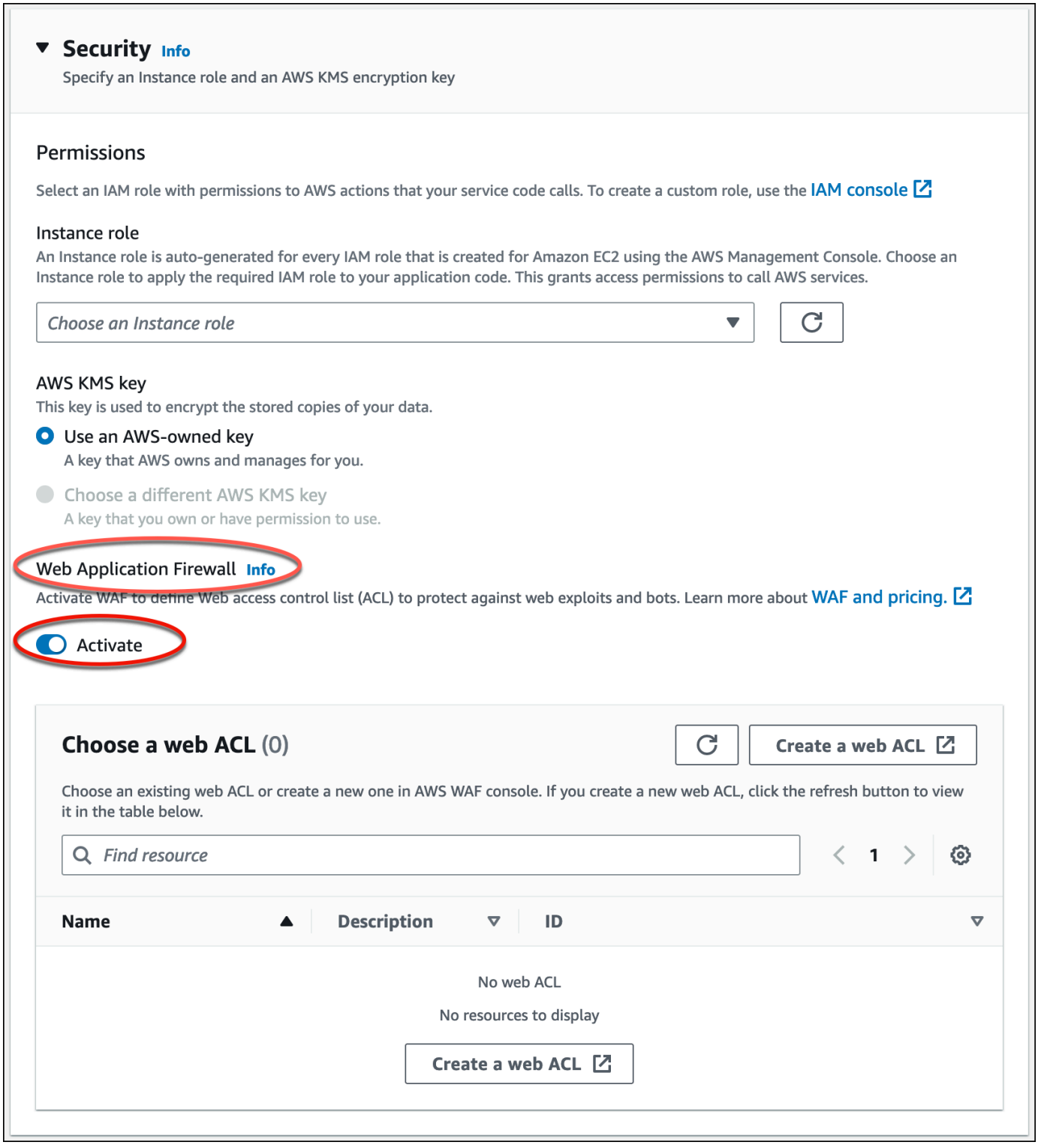
> **Note**
>
> - Un service App Runner ne peut être associé qu'à une seule ACL Web. Cependant, vous pouvez associer une ACL Web à plusieurs services App Runner en plus d'autres AWS ressources.
> - Avant d'associer une ACL Web, assurez-vous de mettre à jour vos autorisations IAM pour AWS WAF. Pour plus d'informations, consultez Autorisations .

Associer une ACL AWS WAF Web

> **Important**
>
> Les règles d'adresse IP source pour les services privés App Runner associés au Web WAF ACLs ne respectent pas les règles basées sur l'adresse IP. Cela est dû au fait que nous ne prenons actuellement pas en charge le transfert des données IP source des demandes vers les services privés App Runner associés à WAF. Si votre application App Runner nécessite des règles de contrôle du trafic IP/CIDR entrant à la source, vous devez utiliser des règles de groupe de sécurité pour les points de terminaison privés plutôt que pour le Web WAF. ACLs

Pour associer une ACL AWS WAF Web

1. Ouvrez la console App Runner, puis dans la liste des régions, sélectionnez votre Région AWS.
2. Selon que vous créez ou mettez à jour un service, effectuez l'une des étapes suivantes :
   - Si vous créez un nouveau service, choisissez Create an App Runner service et accédez à Configurer le service.
   - Si vous mettez à jour un service existant, choisissez l'onglet Configuration, puis sélectionnez Modifier sous Configurer le service.
3. Accédez à Pare-feu d'application Web sous Sécurité.
4. Cliquez sur le bouton Activer pour afficher les options.



5. Effectuez l'une des étapes suivantes :

   - Pour associer une ACL Web existante : Choisissez l'ACL Web requise dans le tableau Choisissez une ACL Web à associer à votre service App Runner.

- Pour créer une nouvelle ACL Web : Choisissez Create Web ACL pour créer une nouvelle ACL Web à l'aide de la AWS WAF console. Pour plus d'informations, consultez la section Création d'une ACL Web dans le manuel du AWS WAF développeur.

  1. Cliquez sur le bouton d'actualisation pour afficher l'ACL Web nouvellement créée dans le tableau Choisissez une ACL Web.
  2. Sélectionnez l'ACL Web requise.

6. Choisissez Suivant si vous créez un nouveau service ou Enregistrer les modifications si vous mettez à jour un service existant. L'ACL Web sélectionné est associé à votre service App Runner.
7. Pour vérifier l'association ACL Web, choisissez l'onglet Configuration de votre service et accédez à Configurer le service. Accédez à Pare-feu d'application Web sous Sécurité pour afficher les détails de l'ACL Web associée à votre service.

> **Note**
>
> Lorsque vous créez une ACL Web, un court laps de temps s'écoule avant que l'ACL Web ne se propage complètement et ne soit disponible pour App Runner. Le temps de propagation peut aller de quelques secondes à plusieurs minutes. AWS WAF renvoie un `WAFUnavailableEntityException` lorsque vous essayez d'associer une ACL Web avant qu'elle ne soit complètement propagée.
> Si vous actualisez le navigateur ou quittez la console App Runner avant que l'ACL Web ne soit complètement propagée, l'association ne se produit pas. Cependant, vous pouvez naviguer dans la console App Runner.

Dissociation d'une ACL AWS WAF Web

Vous pouvez dissocier les AWS WAF sites Web ACl dont vous n'avez plus besoin en mettant à jour votre service App Runner.

Pour dissocier un site Web AWS WAF ACl

1. Ouvrez la console App Runner, puis dans la liste des régions, sélectionnez votre Région AWS.
2. Accédez à l'onglet Configuration du service que vous souhaitez mettre à jour et choisissez Modifier sous Configurer le service.
3. Accédez à Pare-feu d'application Web sous Sécurité.

4. Désactivez le bouton Activer. Vous recevez un message pour confirmer la suppression.
5. Choisissez Confirmer. L'ACL Web est dissociée de votre service App Runner.

> **Note**
>
> - Si vous souhaitez associer votre service à une autre ACL Web, sélectionnez une ACL Web dans le tableau Choisissez une ACL Web. App Runner dissocie l'ACL Web actuel et lance le processus d'association avec l'ACL Web sélectionnée.
> - Si aucun autre service ou ressource App Runner n'utilise une ACL Web dissociée, envisagez de supprimer cette ACL Web. Dans le cas contraire, vous continuerez à encourir des frais. Pour plus d'informations sur la tarification, consultez Tarification d'AWS WAF. Pour obtenir des instructions sur la façon de supprimer une ACL Web, consultez la section DeleteWebACL dans le manuel de référence des AWS WAF API.
> - Vous ne pouvez pas supprimer une ACL Web associée à d'autres services App Runner actifs ou à d'autres ressources.

AWS CLI

Vous pouvez associer ou dissocier une ACL AWS WAF Web en utilisant le AWS WAF public APIs. Le service App Runner, auquel vous souhaitez associer ou dissocier une ACL Web, doit être dans un état valide.

AWS WAF renvoie une `WAFNonexistentItemException` erreur lorsque vous appelez l'une des commandes suivantes AWS WAF APIs pour un service App Runner dont l'état n'est pas valide :

- `AssociateWebACL`
- `DisassociateWebACL`
- `GetWebACLForResource`

Les états non valides pour votre service App Runner sont les suivants :

- `CREATE_FAILED`
- `DELETE_FAILED`
- `DELETED`
- `OPERATION_IN_PROGRESS`

> **Note**
>
> `OPERATION_IN_PROGRESS`L'état n'est pas valide uniquement si votre service App Runner est supprimé.

Pour plus d'informations sur le AWS WAF public APIs, consultez le Guide de référence des AWS WAF API.

> **Note**
>
> Mettez à jour vos autorisations IAM pour AWS WAF. Pour plus d'informations, consultez Autorisations .

Associer une ACL AWS WAF Web à l'aide AWS CLI

> **Important**
>
> Les règles d'adresse IP source pour les services privés App Runner associés au Web WAF ACLs ne respectent pas les règles basées sur l'adresse IP. Cela est dû au fait que nous ne prenons actuellement pas en charge le transfert des données IP source des demandes vers les services privés App Runner associés à WAF. Si votre application App Runner nécessite des règles de contrôle du trafic IP/CIDR entrant à la source, vous devez utiliser des règles de groupe de sécurité pour les points de terminaison privés plutôt que pour le Web WAF. ACLs

Pour associer une ACL AWS WAF Web

1. Créez une ACL AWS WAF Web pour votre service avec votre ensemble préféré d'actions de règles `Allow` ou `Block` de requêtes Web adressées à votre service. Pour plus d'informations AWS WAF APIs, consultez la section CreateWebACL dans le guide de référence des AWS WAF API.

   Example Création d'une ACL Web - Demande

   ```
   aws wafv2
   create-web-acl
   ```

```
--region <region>
--name <web-acl-name>
--scope REGIONAL
--default-action Allow={}
--visibility-config <file-name.json>
# This is the file containing the WAF web ACL rules.
```

2. Associez l'ACL Web que vous avez créée au service App Runner à l'aide de l'API `associate-web-acl` AWS WAF publique. Pour plus d'informations AWS WAF APIs, consultez la section AssociateWebACL dans le guide de référence des AWS WAF API.

   > **Note**
   >
   > Lorsque vous créez une ACL Web, un court laps de temps s'écoule avant que l'ACL Web ne se propage complètement et ne soit disponible pour App Runner. Le temps de propagation peut aller de quelques secondes à plusieurs minutes. AWS WAF renvoie un `WAFUnavailableEntityException` lorsque vous essayez d'associer une ACL Web avant qu'elle ne soit complètement propagée.
   > Si vous actualisez le navigateur ou quittez la console App Runner avant que l'ACL Web ne soit complètement propagée, l'association ne se produit pas. Cependant, vous pouvez naviguer dans la console App Runner.

   Example Associer une ACL Web - Requête

   ```
   aws wafv2 associate-web-acl
   --resource-arn <apprunner_service_arn>
   --web-acl-arn <web_acl_arn>
   --region <region>
   ```

3. Vérifiez que l'ACL Web est associée à votre service App Runner à l'aide de l'API `get-web-acl-for-resource` AWS WAF publique. Pour plus d'informations AWS WAF APIs, consultez la section GetWebACLForRessource du guide de référence des AWS WAF API.

   Example Vérifier l'ACL Web pour la ressource - Demande

   ```
   aws wafv2 get-web-acl-for-resource
   --resource-arn <apprunner_service_arn>
   --region <region>
   ```

Si aucun site Web n'est ACLs associé à votre service, vous recevez une réponse vide.

Suppression d'une ACL AWS WAF Web à l'aide de AWS CLI

Vous ne pouvez pas supprimer une ACL AWS WAF Web si elle est associée à un service App Runner.

Pour supprimer une ACL AWS WAF Web

1. Dissociez l'ACL Web de votre service App Runner à l'aide de l'API `disassociate-web-acl` AWS WAF publique. Pour plus d'informations AWS WAF APIs, consultez la section DisassociateWebACL dans le guide de référence des AWS WAF API.

   Example Dissociation d'une ACL Web - Demande

   ```
   aws wafv2 disassociate-web-acl
   --resource-arn <apprunner_service_arn>
   --region <region>
   ```

2. Vérifiez que l'ACL Web est dissociée de votre service App Runner à l'aide de l'API `get-web-acl-for-resource` AWS WAF publique.

   Example Vérifiez que l'ACL Web est dissociée - Demande

   ```
   aws wafv2 get-web-acl-for-resource
   --resource-arn <apprunner_service_arn>
   --region <region>
   ```

   L'ACL Web dissociée n'est pas répertoriée pour votre service App Runner. Si aucun site Web n'est ACLs associé à votre service, vous recevez une réponse vide.

3. Supprimez l'ACL Web dissociée à l'aide de l'API `delete-web-acl` AWS WAF publique. Pour plus d'informations AWS WAF APIs, consultez la section DeleteWebACL dans le guide de référence des AWS WAF API.

   Example Supprimer une ACL Web - Demande

   ```
   aws wafv2 delete-web-acl
   --name <web_acl_name>
   --scope REGIONAL
   ```

```
--id <web_acl_id>
--lock-token <web_acl_lock_token>
--region <region>
```

4. Vérifiez que l'ACL Web est supprimée à l'aide de l'API `list-web-acl` AWS WAF publique. Pour plus d'informations à ce sujet AWS WAF APIs, consultez ListWebACLsle Guide de référence des AWS WAF API.

   Example Vérifiez que l'ACL Web est supprimée - Demande

   ```
   aws wafv2 list-web-acls
   --scope REGIONAL
   --region <region>
   ```

   L'ACL Web supprimée ne figure plus dans la liste.

   > **Note**
   >
   > Si une ACL Web est associée à d'autres services App Runner actifs ou à d'autres ressources, telles que des groupes d'utilisateurs Amazon Cognito, l'ACL Web ne peut pas être supprimée.

Répertorier les services App Runner associés à une ACL Web

Une ACL Web peut être associée à plusieurs services App Runner et à d'autres ressources. Répertoriez les services App Runner associés à une ACL Web à l'aide de l'API `list-resources-for-web-acl` AWS WAF publique. Pour plus d'informations AWS WAF APIs, consultez la section ListResourcesForWebACL dans le guide de référence des AWS WAF API.

Example Lister les services App Runner associés à une ACL Web - Demande

```
aws wafv2 list-resources-for-web-acl
--web-acl-arn <WEB_ACL_ARN>
--resource-type APP_RUNNER_SERVICE
--region <REGION>
```

Example Lister les services App Runner associés à une ACL Web - Response

L'exemple suivant illustre la réponse lorsqu'aucun service App Runner n'est associé à une ACL Web.

```
{
  "ResourceArns": []
}
```

Example Lister les services App Runner associés à une ACL Web - Response

L'exemple suivant illustre la réponse lorsque des services App Runner sont associés à une ACL Web.

```
{
  "ResourceArns": [
    "arn:aws:apprunner:<region>:<aws_account_id>:service/<service_name>/<service_id>"
  ]
}
```

Tester et enregistrer sur AWS WAF le Web ACLs

Lorsque vous définissez une action de règle sur Count dans votre ACL Web AWS WAF , vous ajoutez la demande au nombre de demandes correspondant à la règle. Pour tester une ACL Web avec votre service App Runner, définissez les actions des règles sur Count et prenez en compte le volume de demandes correspondant à chaque règle. Par exemple, vous définissez une règle pour l'`Block`action qui correspond à un grand nombre de demandes que vous considérez comme relevant du trafic utilisateur normal. Dans ce cas, vous devrez peut-être reconfigurer votre règle. Pour plus d'informations, consultez la section Tester et régler vos AWS WAF protections dans le Guide du AWS WAF développeur.

Vous pouvez également configurer AWS WAF pour consigner les en-têtes des demandes dans un groupe de CloudWatch journaux Amazon Logs, un bucket Amazon Simple Storage Service (Amazon S3) ou un Amazon Data Firehose. Pour plus d’informations, consultez Journalisation du trafic ACL web dans le Guide du développeur AWS WAF .

Pour accéder aux journaux liés à l'ACL Web associée à votre service App Runner, reportez-vous aux champs de journal suivants :

- `httpSourceName`: Contient `APPRUNNER`
- `httpSourceId`: Contient `customeraccountid-apprunnerserviceid`

Pour plus d'informations, consultez la section Exemples de journaux dans le guide du AWS WAF développeur.

> **Important**
>
> Les règles d'adresse IP source pour les services privés App Runner associés au Web WAF ACLs ne respectent pas les règles basées sur l'adresse IP. Cela est dû au fait que nous ne prenons actuellement pas en charge le transfert des données IP source des demandes vers les services privés App Runner associés à WAF. Si votre application App Runner nécessite des règles de contrôle du trafic IP/CIDR entrant à la source, vous devez utiliser des règles de groupe de sécurité pour les points de terminaison privés plutôt que pour le Web WAF. ACLs

Configuration des options du service App Runner à l'aide d'un fichier de configuration

> **Note**
>
> Les fichiers de configuration s'appliquent uniquement aux services basés sur le code source. Vous ne pouvez pas utiliser de fichiers de configuration avec des services basés sur des images.

Lorsque vous créez un AWS App Runner service à l'aide d'un référentiel de code source, vous avez AWS App Runner besoin d'informations sur la création et le démarrage de votre service. Vous pouvez fournir ces informations chaque fois que vous créez un service à l'aide de la console ou de l'API App Runner. Vous pouvez également définir les options de service à l'aide d'un fichier de configuration. Les options que vous spécifiez dans un fichier font partie de votre référentiel source, et toutes les modifications apportées à ces options sont suivies de la même manière que les modifications apportées au code source. Vous pouvez utiliser le fichier de configuration d'App Runner pour spécifier plus d'options que celles prises en charge par l'API. Il n'est pas nécessaire de fournir un fichier de configuration si vous n'avez besoin que des options de base prises en charge par l'API.

Le fichier de configuration App Runner est un fichier YAML nommé `apprunner.yaml` dans le répertoire source du référentiel de votre application. Il fournit des options de compilation et d'exécution pour votre service. Les valeurs de ce fichier indiquent à App Runner comment créer et démarrer votre service, et fournissent un contexte d'exécution tel que les paramètres réseau et les variables d'environnement.

Le fichier de configuration d'App Runner n'inclut pas les paramètres opérationnels, tels que le processeur et la mémoire.

Pour des exemples de fichiers de configuration d'App Runner, consultezthe section called “Exemples”. Pour un guide de référence complet, voirthe section called “Référence”.

Rubriques

- Exemples de fichiers de configuration d'App Runner
- Référence du fichier de configuration App Runner

Exemples de fichiers de configuration d'App Runner

> **Note**
>
> Les fichiers de configuration s'appliquent uniquement aux services basés sur le code source. Vous ne pouvez pas utiliser de fichiers de configuration avec des services basés sur des images.

Les exemples suivants illustrent les fichiers AWS App Runner de configuration. Certains sont minimaux et contiennent uniquement les paramètres requis. D'autres sont complets, y compris toutes les sections du fichier de configuration. Pour un aperçu des fichiers de configuration d'App Runner, consultezFichier de configuration d'App Runner.

Exemples de fichiers de configuration

Fichier de configuration minimal

Avec un fichier de configuration minimal, App Runner part des hypothèses suivantes :

- Aucune variable d'environnement personnalisée n'est nécessaire lors de la construction ou de l'exécution.
- La dernière version d'exécution est utilisée.
- Le numéro de port par défaut et la variable d'environnement de port sont utilisés.

Example apprunner.yaml

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

Fichier de configuration complet

Cet exemple montre l'utilisation de toutes les clés de configuration dans le format `apprunner.yaml` d'origine avec un environnement d'exécution géré.

Example apprunner.yaml

```
version: 1.0
runtime: python3
build:
  commands:
    pre-build:
      - wget -c https://s3.amazonaws.com/amzn-s3-demo-bucket/test-lib.tar.gz -O - | tar
 -xz
    build:
      - pip install pipenv
      - pipenv install
    post-build:
      - python manage.py test
  env:
    - name: DJANGO_SETTINGS_MODULE
      value: "django_apprunner.settings"
    - name: MY_VAR_EXAMPLE
      value: "example"
run:
  runtime-version: 3.7.7
  command: pipenv run gunicorn django_apprunner.wsgi --log-file -
  network:
    port: 8000
    env: MY_APP_PORT
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
  secrets:
    - name: my-secret
      value-from: "arn:aws:secretsmanager:us-
east-1:123456789012:secret:testingstackAppRunnerConstr-kJFXde2ULKbT-S7t8xR:username::"
    - name: my-parameter
      value-from: "arn:aws:ssm:us-east-1:123456789012:parameter/parameter-name"
    - name: my-parameter-only-name
      value-from: "parameter-name"
```

Fichier de configuration complet — (utilise une version révisée)

Cet exemple montre l'utilisation de toutes les clés de configuration dans `apprunner.yaml` un environnement d'exécution géré.

Le `pre-run` paramètre n'est pris en charge que par la version révisée d'App Runner. N'insérez pas ce paramètre dans votre fichier de configuration si votre application utilise des versions d'exécution prises en charge par la version originale d'App Runner. Pour de plus amples informations, veuillez consulter Versions d'exécution gérées et build d'App Runner.

> **Note**
>
> Comme cet exemple concerne Python 3.11, nous utilisons les `python3` commandes `pip3` et. Pour plus d'informations, consultez la rubrique relative Des légendes pour des versions d'exécution spécifiques à la plateforme Python.

Example apprunner.yaml

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

```
  command: pipenv run gunicorn django_apprunner.wsgi --log-file -
  network:
    port: 8000
    env: MY_APP_PORT
  env:
    - name: MY_VAR_EXAMPLE
      value: "example"
  secrets:
    - name: my-secret
      value-from: "arn:aws:secretsmanager:us-
east-1:123456789012:secret:testingstackAppRunnerConstr-kJFXde2ULKbT-S7t8xR:username::"
    - name: my-parameter
      value-from: "arn:aws:ssm:us-east-1:123456789012:parameter/parameter-name"
    - name: my-parameter-only-name
      value-from: "parameter-name"
```

Pour des exemples de fichiers de configuration d'exécution gérés spécifiques, consultez la sous-rubrique spécifique à l'exécution sousService basé sur le code.

Référence du fichier de configuration App Runner

> **Note**
>
> Les fichiers de configuration s'appliquent uniquement aux services basés sur le code source. Vous ne pouvez pas utiliser de fichiers de configuration avec des services basés sur des images.

Cette rubrique est un guide de référence complet sur la syntaxe et la sémantique d'un fichier de AWS App Runner configuration. Pour un aperçu des fichiers de configuration d'App Runner, consultezFichier de configuration d'App Runner.

Le fichier de configuration d'App Runner est un fichier YAML. Nommez-le `apprunner.yaml` et placez-le dans le répertoire source du référentiel de votre application.

Aperçu de la structure

Le fichier de configuration d'App Runner est un fichier YAML. Nommez-le `apprunner.yaml` et placez-le dans le répertoire source du référentiel de votre application.

Le fichier de configuration d'App Runner contient les éléments principaux suivants :

- Section supérieure — Contient les clés de niveau supérieur
- Section de construction — Configure la phase de construction
- Section Exécuter — Configure la phase d'exécution

Section supérieure

Les clés situées en haut du fichier fournissent des informations générales sur le fichier et l'exécution de votre service. Les touches disponibles sont les suivantes :

- `version`— Obligatoire. Version du fichier de configuration d'App Runner. Dans l'idéal, utilisez la dernière version.

  Syntaxe

  ```
  version: version
  ```

  Example

  ```
  version: 1.0
  ```

- `runtime`— Obligatoire. Nom du moteur d'exécution utilisé par votre application. Pour en savoir plus sur les temps d'exécution disponibles pour les différentes plateformes de programmation proposées par App Runner, consultezService basé sur le code.

  > **Note**
  >
  > La convention de dénomination d'un environnement d'exécution géré est*<language-name><major-version>*.

  Syntaxe

  ```
  runtime: runtime-name
  ```

  Example

  ```
  runtime: python3
  ```

Section de construction

La section build configure la phase de compilation du déploiement du service App Runner. Vous pouvez spécifier des commandes de construction et des variables d'environnement. Les commandes de construction sont obligatoires.

La section commence par la `build:` clé et comporte les sous-clés suivantes :

- `commands`— Obligatoire. Spécifie les commandes exécutées par App Runner au cours des différentes phases de construction. Inclut les sous-clés suivantes :
  - `pre-build`— Facultatif. Les commandes exécutées par App Runner avant la compilation. Par exemple, installez des npm dépendances ou testez des bibliothèques.
  - `build`— Obligatoire. Les commandes qu'App Runner exécute pour créer votre application. Par exemple, utilisezpipenv.
  - `post-build`— Facultatif. Les commandes exécutées par App Runner après la compilation. Par exemple, utilisez Maven pour empaqueter des artefacts de construction dans un fichier JAR ou WAR, ou pour exécuter un test.

  Syntaxe

```
build:
  commands:
    pre-build:
      - command
      - …
    build:
      - command
      - …
    post-build:
      - command
      - …
```

  Example

```
build:
  commands:
    pre-build:
      - yum install openssl
    build:
      - pip install -r requirements.txt
```

```
    post-build:
      - python manage.py test
```

- `env`— Facultatif. Spécifie des variables d'environnement personnalisées pour la phase de construction. Défini comme des mappages scalaires nom-valeur. Vous pouvez faire référence à ces variables par leur nom dans vos commandes de compilation.

> **Note**
>
> Il existe deux `env` entrées distinctes situées à deux emplacements différents dans ce fichier de configuration. Un ensemble se trouve dans la section Build et l'autre dans la section Run.
>
> - L'`env`ensemble de la section Build peut être référencé par les `pre-run` commandes `pre-build` `build``post-build`,, et pendant le processus de génération.
>
> Important - Notez que les `pre-run` commandes se trouvent dans la section Exécuter de ce fichier, même si elles ne peuvent accéder qu'aux variables d'environnement définies dans la section Build.
> - L'`env`ensemble défini dans la section Exécuter peut être référencé par la `run` commande dans l'environnement d'exécution.

Syntaxe

```
build:
  env:
    - name: name1
      value: value1
    - name: name2
      value: value2
    - …
```

Example

```
build:
  env:
    - name: DJANGO_SETTINGS_MODULE
      value: "django_apprunner.settings"
    - name: MY_VAR_EXAMPLE
```

```
      value: "example"
```

Exécuter la section

La section d'exécution configure la phase d'exécution du conteneur dans le cadre du déploiement de l'application App Runner. Vous pouvez spécifier la version d'exécution, les commandes de pré-exécution (format révisé uniquement), la commande de démarrage, le port réseau et les variables d'environnement.

La section commence par la `run:` clé et comporte les sous-clés suivantes :

- `runtime-version`— Facultatif. Spécifie la version d'exécution que vous souhaitez verrouiller pour votre service App Runner.

  Par défaut, seule la version majeure est verrouillée. App Runner utilise les dernières versions mineures et correctifs disponibles pour l'exécution lors de chaque déploiement ou mise à jour de service. Si vous spécifiez des versions majeures et mineures, les deux sont verrouillées et App Runner ne met à jour que les versions de correctif. Si vous spécifiez des versions majeures, mineures et patchs, votre service est verrouillé sur une version d'exécution spécifique et App Runner ne la met jamais à jour.

  Syntaxe

  ```
  run:
    runtime-version: major[.minor[.patch]]
  ```

  > **Note**
  >
  > Les environnements d'exécution de certaines plateformes ont des composants de version différents. Consultez les rubriques spécifiques à la plateforme pour plus de détails.

  Example

  ```
  runtime: python3
  run:
    runtime-version: 3.7
  ```

- `pre-run`— Facultatif. Utilisation de la version révisée uniquement. Spécifie les commandes qu'App Runner exécute après avoir copié votre application de l'image de compilation vers l'image d'exécution. Vous pouvez entrer des commandes ici pour modifier l'image exécutée en dehors du `/app` répertoire. Par exemple, si vous devez installer des dépendances globales supplémentaires résidant en dehors du `/app` répertoire, entrez les commandes requises dans cette sous-section pour ce faire. Pour plus d'informations sur le processus de création d'App Runner, consultezVersions d'exécution gérées et build d'App Runner.

  > **Note**
  >
  > - Important — Même si les `pre-run` commandes sont répertoriées dans la section Exécuter, elles ne peuvent faire référence qu'aux variables d'environnement définies dans la section Build de ce fichier de configuration. Ils ne peuvent pas faire référence aux variables d'environnement définies dans cette section Exécuter.
  > - Le `pre-run` paramètre n'est pris en charge que par la version révisée d'App Runner. N'insérez pas ce paramètre dans votre fichier de configuration si votre application utilise des versions d'exécution prises en charge par la version originale d'App Runner. Pour de plus amples informations, veuillez consulter Versions d'exécution gérées et build d'App Runner.

  Syntaxe

  ```
  run:
    pre-run:
        - command
        - …
  ```

- `command`— Obligatoire. Commande utilisée par App Runner pour exécuter votre application une fois la création de l'application terminée.

  Syntaxe

  ```
  run:
    command: command
  ```

- `network`— Facultatif. Spécifie le port que votre application écoute. Elle comprend les éléments suivants :

- `port`— Facultatif. S'il est spécifié, il s'agit du numéro de port que votre application écoute. La valeur par défaut est `8080`.
- `env`— Facultatif. Si cela est spécifié, App Runner transmet le numéro de port au conteneur dans cette variable d'environnement, en plus de transmettre (et non à la place) le même numéro de port dans la variable d'environnement par défaut,`PORT`. En d'autres termes, si vous le spécifiez`env`, App Runner transmet le numéro de port dans deux variables d'environnement.

Syntaxe

```
run:
  network:
    port: port-number
    env: env-variable-name
```

Example

```
run:
  network:
    port: 8000
    env: MY_APP_PORT
```

- `env`— Facultatif. Définition de variables d'environnement personnalisées pour la phase d'exécution. Défini comme des mappages scalaires nom-valeur. Vous pouvez faire référence à ces variables par leur nom dans votre environnement d'exécution.

> **Note**
>
> Il existe deux `env` entrées distinctes situées à deux emplacements différents dans ce fichier de configuration. Un ensemble se trouve dans la section Build et l'autre dans la section Run.
>
> - L'`env`ensemble de la section Build peut être référencé par les `pre-run` commandes `pre-build` `build`,`post-build`, et pendant le processus de génération.
>
> Important - Notez que les `pre-run` commandes se trouvent dans la section Exécuter de ce fichier, même si elles ne peuvent accéder qu'aux variables d'environnement définies dans la section Build.

- L'`env`ensemble défini dans la section Exécuter peut être référencé par la `run` commande dans l'environnement d'exécution.

Syntaxe

```
run:
  env:
    - name: name1
      value: value1
    - name: name2
      value: value2
  secrets:
    - name: name1
      value-from: arn:aws:secretsmanager:region:aws_account_id:secret:secret-id
    - name: name2
      value-from: arn:aws:ssm:region:aws_account_id:parameter/parameter-name
    - …
```

Example

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

L'API App Runner

L'interface de programmation d' AWS App Runner applications (API) est une RESTful API permettant d'envoyer des requêtes au service App Runner. Vous pouvez utiliser l'API pour créer, répertorier, décrire, mettre à jour et supprimer des ressources App Runner dans votre Compte AWS.

Vous pouvez appeler l'API directement dans le code de votre application ou utiliser l'une des AWS SDKs.

Pour obtenir des informations de référence complètes sur les API, consultez la référence des AWS App Runner API.

Pour plus d'informations sur les outils de AWS développement, voir Outils sur lesquels s'appuyer AWS.

Rubriques

- Utilisation du AWS CLI pour travailler avec App Runner
- Utiliser AWS CloudShell pour travailler avec AWS App Runner

Utilisation du AWS CLI pour travailler avec App Runner

Pour les scripts de ligne de commande, utilisez le AWS CLIpour appeler le service App Runner. Pour obtenir des informations AWS CLI de référence complètes, consultez l'apprunner dans le manuel de référence des AWS CLI commandes.

AWS CloudShell vous permet de ne pas installer le AWS CLI dans votre environnement de développement et de l'utiliser à la AWS Management Console place. En plus d'éviter l'installation, vous n'avez pas besoin de configurer les informations d'identification, ni de spécifier la région. Votre AWS Management Console session fournit ce contexte au AWS CLI. Pour plus d'informations CloudShell et pour un exemple d'utilisation, consultezthe section called “En utilisant AWS CloudShell”.

Utiliser AWS CloudShell pour travailler avec AWS App Runner

AWS CloudShell est un shell pré-authentifié basé sur un navigateur que vous pouvez lancer directement depuis le. AWS Management Console Vous pouvez exécuter des AWS CLI commandes

sur AWS des services (y compris AWS App Runner) à l'aide de votre shell préféré (Bash PowerShell ou Z shell). Et vous pouvez le faire sans télécharger ou installer des outils de ligne de commande.

Vous lancez AWS CloudShell à partir de AWS Management Console, et les AWS informations d'identification que vous avez utilisées pour vous connecter à la console sont automatiquement disponibles dans une nouvelle session shell. Cette pré-authentification des AWS CloudShell utilisateurs vous permet d'ignorer la configuration des informations d'identification lorsque vous interagissez avec AWS des services tels qu'App Runner à l'aide de la AWS CLI version 2 (préinstallée sur l'environnement informatique du shell).

Rubriques

- Obtention des autorisations IAM pour AWS CloudShell
- Interaction avec App Runner à l'aide de AWS CloudShell
- Vérification de votre service App Runner à l'aide de AWS CloudShell

Obtention des autorisations IAM pour AWS CloudShell

À l'aide des ressources de gestion des accès fournies par Gestion des identités et des accès AWS, les administrateurs peuvent accorder des autorisations aux utilisateurs IAM afin qu'ils puissent accéder aux fonctionnalités de l'environnement AWS CloudShell et les utiliser.

Le moyen le plus rapide pour un administrateur d'accorder l'accès aux utilisateurs est d'utiliser une politique AWS gérée. Une politique gérée par AWS est une politique autonome qui est créée et gérée par AWS. La politique AWS gérée suivante pour CloudShell peut être attachée aux identités IAM :

- `AWSCloudShellFullAccess`: accorde l'autorisation d'utilisation AWS CloudShell avec un accès complet à toutes les fonctionnalités.

Si vous souhaitez limiter l'étendue des actions qu'un utilisateur IAM peut effectuer AWS CloudShell, vous pouvez créer une politique personnalisée qui utilise la stratégie `AWSCloudShellFullAccess` gérée comme modèle. Pour plus d'informations sur la limitation des actions disponibles pour les utilisateurs dans CloudShell, consultez la section Gestion de l' AWS CloudShell accès et de l'utilisation avec les politiques IAM dans le Guide de l'AWS CloudShell utilisateur.

> **Note**
>
> Votre identité IAM nécessite également une politique autorisant App Runner à passer des appels. Pour de plus amples informations, veuillez consulter the section called “App Runner et IAM”.

Interaction avec App Runner à l'aide de AWS CloudShell

Après le lancement AWS CloudShell depuis le AWS Management Console, vous pouvez immédiatement commencer à interagir avec App Runner à l'aide de l'interface de ligne de commande.

Dans l'exemple suivant, vous récupérez des informations sur l'un de vos services App Runner à l'aide du AWS CLI in CloudShell.

> **Note**
>
> Lorsque vous AWS CLI l'utilisez AWS CloudShell, vous n'avez pas besoin de télécharger ou d'installer de ressources supplémentaires. De plus, comme vous êtes déjà authentifié dans le shell, vous n'avez pas besoin de configurer les informations d'identification avant d'effectuer des appels.

Example Récupération des informations de service App Runner à l'aide de AWS CloudShell

1. À partir de AWS Management Console, vous pouvez lancer CloudShell en choisissant les options suivantes disponibles dans la barre de navigation :
   - Choisissez l' CloudShell icône.
   - Commencez **`cloudshell`** à taper dans le champ de recherche, puis choisissez l'CloudShelloption lorsqu'elle apparaît dans les résultats de recherche.
2. Pour répertorier tous les services App Runner actuels de votre AWS compte dans la AWS région de la session de console, entrez la commande suivante dans la ligne de CloudShell commande :

   ```
   $ aws apprunner list-services
   ```

   La sortie répertorie les informations récapitulatives relatives à vos services.

```
{
  "ServiceSummaryList": [
    {
      "ServiceName": "my-app-1",
      "ServiceId": "8fe1e10304f84fd2b0df550fe98a71fa",
      "ServiceArn": "arn:aws:apprunner:us-east-2:123456789012:service/my-
app-1/8fe1e10304f84fd2b0df550fe98a71fa",
      "ServiceUrl": "psbqam834h.us-east-1.awsapprunner.com",
      "CreatedAt": "2020-11-20T19:05:25Z",
      "UpdatedAt": "2020-11-23T12:41:37Z",
      "Status": "RUNNING"
    },
    {
      "ServiceName": "my-app-2",
      "ServiceId": "ab8f94cfe29a460fb8760afd2ee87555",
      "ServiceArn": "arn:aws:apprunner:us-east-2:123456789012:service/my-app-2/
ab8f94cfe29a460fb8760afd2ee87555",
      "ServiceUrl": "e2m8rrrx33.us-east-1.awsapprunner.com",
      "CreatedAt": "2020-11-06T23:15:30Z",
      "UpdatedAt": "2020-11-23T13:21:22Z",
      "Status": "RUNNING"
    }
  ]
}
```

3. Pour obtenir une description détaillée d'un service App Runner en particulier, entrez la commande suivante dans la ligne de CloudShell commande, en utilisant l'une des commandes ARNs récupérées à l'étape précédente :

```
$ aws apprunner describe-service --service-arn arn:aws:apprunner:us-
east-2:123456789012:service/my-app-1/8fe1e10304f84fd2b0df550fe98a71fa
```

La sortie contient une description détaillée du service que vous avez spécifié.

```
{
  "Service": {
    "ServiceName": "my-app-1",
    "ServiceId": "8fe1e10304f84fd2b0df550fe98a71fa",
    "ServiceArn": "arn:aws:apprunner:us-east-2:123456789012:service/my-
app-1/8fe1e10304f84fd2b0df550fe98a71fa",
    "ServiceUrl": "psbqam834h.us-east-1.awsapprunner.com",
```

```
    "CreatedAt": "2020-11-20T19:05:25Z",
    "UpdatedAt": "2020-11-23T12:41:37Z",
    "Status": "RUNNING",
    "SourceConfiguration": {
      "CodeRepository": {
        "RepositoryUrl": "https://github.com/my-account/python-hello",
        "SourceCodeVersion": {
          "Type": "BRANCH",
          "Value": "main"
        },
        "CodeConfiguration": {
          "CodeConfigurationValues": {
            "BuildCommand": "[pip install -r requirements.txt]",
            "Port": "8080",
            "Runtime": "PYTHON_3",
            "RuntimeEnvironmentVariables": [
              {
                "NAME": "Jane"
              }
            ],
            "StartCommand": "python server.py"
          },
          "ConfigurationSource": "API"
        }
      },
      "AutoDeploymentsEnabled": true,
      "AuthenticationConfiguration": {
        "ConnectionArn": "arn:aws:apprunner:us-east-2:123456789012:connection/my-
github-connection/e7656250f67242d7819feade6800f59e"
      }
    },
    "InstanceConfiguration": {
      "CPU": "1 vCPU",
      "Memory": "3 GB"
    },
    "HealthCheckConfiguration": {
      "Protocol": "TCP",
      "Path": "/",
      "Interval": 10,
      "Timeout": 5,
      "HealthyThreshold": 1,
      "UnhealthyThreshold": 5
    },
    "AutoScalingConfigurationSummary": {
```

```
      "AutoScalingConfigurationArn": "arn:aws:apprunner:us-
east-2:123456789012:autoscalingconfiguration/
DefaultConfiguration/1/00000000000000000000000000000001",
      "AutoScalingConfigurationName": "DefaultConfiguration",
      "AutoScalingConfigurationRevision": 1
    }
  }
}
```

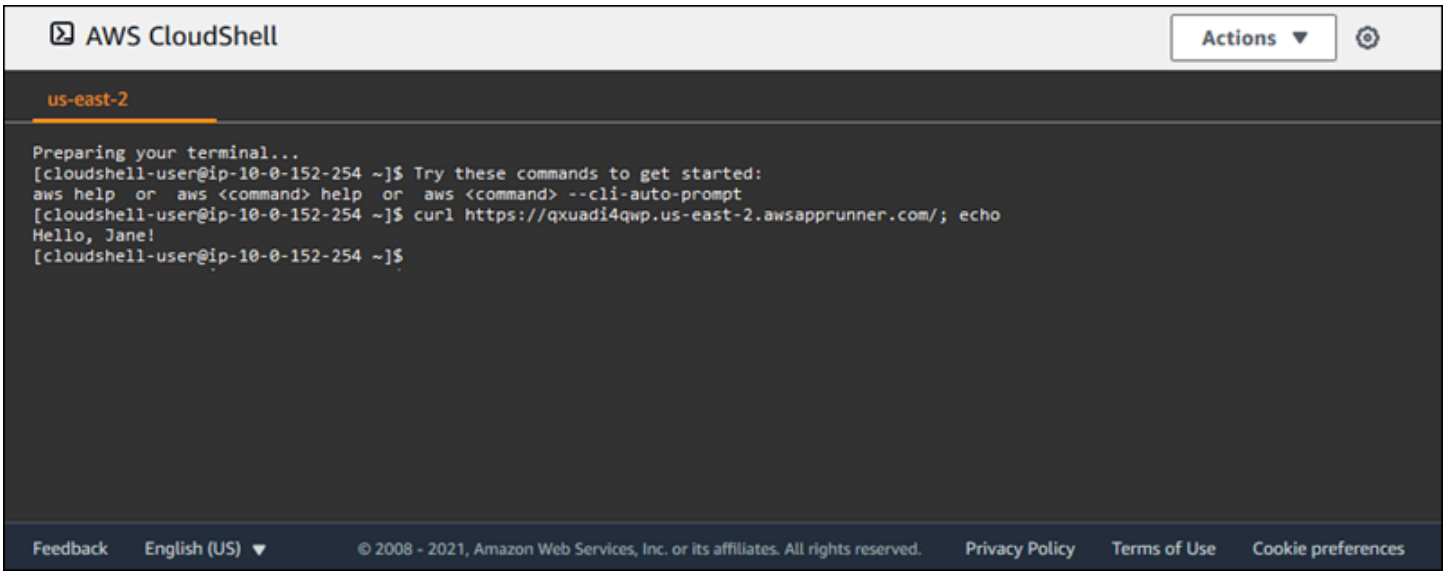
Vérification de votre service App Runner à l'aide de AWS CloudShell

Lorsque vous créez un service App Runner, App Runner crée un domaine par défaut pour le site Web de votre service et l'affiche dans la console (ou le renvoie dans le résultat de l'appel d'API). Vous pouvez l'utiliser CloudShell pour passer des appels vers votre site Web et vérifier qu'il fonctionne correctement.

Par exemple, après avoir créé un service App Runner comme décrit dansPrise en main, exécutez la commande suivante dans CloudShell :

```
$ curl https://qxuadi4qwp.us-east-2.awsapprunner.com/; echo
```

La sortie doit afficher le contenu de page attendu.



Résolution des problèmes

Ce chapitre décrit les étapes de résolution des erreurs et problèmes courants que vous pouvez rencontrer lors de l'utilisation de votre AWS App Runner service. Les messages d'erreur peuvent apparaître sur la console, l'API ou l'onglet Logs de votre page de service.

Pour plus de conseils de dépannage et de réponses aux questions courantes de support, visitez le Centre de connaissances .

Rubriques

- Lorsque le service ne parvient pas à créer
- noms de domaine personnalisés
- Erreur de routage des demandes HTTP/HTTPS
- Lorsque le service ne parvient pas à se connecter à Amazon RDS ou au service en aval
- Lorsqu'il n'y a pas assez d'adresses IP pour le lancement d'instances ou le dimensionnement

Lorsque le service ne parvient pas à créer

Si votre tentative de création d'un service App Runner échoue, le service passe à un `CREATE_FAILED` état. Cet état indique que la création a échoué sur la console. La création d'un service peut échouer en raison de problèmes liés à un ou plusieurs des éléments suivants :

- Le code de votre application
- Le processus de construction
- Configuration
- Quotas de ressources
- Problèmes temporaires liés au Services AWS sous-jacent utilisé par votre service

Pour résoudre les problèmes liés à l'échec de la création d'un service, nous vous recommandons de procéder comme suit.

1. Lisez les événements et les journaux du service pour découvrir pourquoi le service n'a pas pu être créé.
2. Apportez les modifications nécessaires à votre code ou à votre configuration.

3. Si vous avez atteint votre quota de services, supprimez un ou plusieurs services.
4. Si vous avez atteint un autre quota de ressources, vous pourrez peut-être l'augmenter s'il est ajustable.
5. Réessayez de rétablir le service après avoir effectué toutes les étapes ci-dessus. Pour plus d'informations sur la façon de reconstruire votre service, consultezthe section called “Reconstruire le service défaillant”.

> **Note**
>
> L'un des quotas de ressources ajustables susceptibles de poser problème est la ressource vCPU Fargate On-Demand.
> Le nombre de ressources vCPU détermine le nombre d'instances qu'App Runner peut fournir à votre service. Il s'agit d'une valeur de quota ajustable pour le nombre de ressources de vCPU Fargate On-Demand résidant dans le service. AWS Fargate Pour consulter les paramètres de quota de vCPU de votre compte ou pour demander une augmentation de quota, utilisez la console Service Quotas dans le. AWS Management Console Pour plus d'informations, consultez les quotas AWS Fargate de service dans le manuel Amazon Elastic Container Service Developer Guide.

> **Important**
>
> Aucun frais supplémentaire n'est facturé au-delà de la tentative de création initiale en cas d'échec d'un service. Même si le service défaillant n'est pas utilisable, il est toujours pris en compte dans votre quota de service. App Runner ne supprime pas automatiquement le service défaillant. Assurez-vous donc de le supprimer lorsque vous aurez terminé d'analyser l'échec.

noms de domaine personnalisés

Cette section explique comment résoudre les différentes erreurs que vous pourriez rencontrer lors de la création d'un lien vers un domaine personnalisé.

> **Note**
>
> Pour renforcer la sécurité de vos applications App Runner, le domaine*.awsapprunner.com est enregistré dans la liste des suffixes publics (PSL). Pour plus de sécurité, nous vous recommandons d'utiliser des cookies avec un `__Host-` préfixe si vous devez définir des cookies sensibles dans le nom de domaine par défaut de vos applications App Runner. Cette pratique vous aidera à protéger votre domaine contre les tentatives de falsification de requêtes intersites (CSRF). Pour plus d'informations, consultez la page Set-Cookie du Mozilla Developer Network.

Obtention d'une erreur Create Fail pour un domaine personnalisé

- Vérifiez si cette erreur est due à un problème avec les enregistrements CAA. S'il n'y a aucun enregistrement CAA dans l'arborescence DNS, vous recevez un message `fail open` et vous AWS Certificate Manager émettez un certificat pour vérifier le domaine personnalisé. Cela permet à App Runner d'accepter le domaine personnalisé. Si vous utilisez des certifications CAA dans les enregistrements DNS, assurez-vous qu'au moins les enregistrements CAA d'un domaine incluent`amazon.com`. Dans le cas contraire, ACM ne pourra pas émettre de certificat. Par conséquent, le domaine personnalisé pour App Runner ne peut pas être créé.

  L'exemple suivant utilise l'outil de recherche DNS DiG pour afficher les enregistrements CAA auxquels il manque une entrée obligatoire. L'exemple utilise `example.com` comme domaine personnalisé. Exécutez les commandes suivantes dans l'exemple pour vérifier les enregistrements CAA.

  ```
  ...
  ;; QUESTION SECTION:
  ;example.com.           IN  CAA

  ;; ANSWER SECTION:
  example.com.        7200    IN  CAA 0 iodef "mailto:hostmaster@example.com"
  example.com.        7200    IN  CAA 0 issue "letsencrypt.org"
  ...note absence of "amazon.com" in any of the above CAA records...
  ```

- Corrigez les enregistrements de domaine et assurez-vous qu'au moins un enregistrement CAA inclut`amazon.com`.
- Réessayez de lier le domaine personnalisé à App Runner.

Pour obtenir des instructions sur la façon de résoudre les erreurs CAA, consultez les rubriques suivantes :

- Problèmes liés à l'autorisation de l'autorité de certification (CAA)
- Comment résoudre les erreurs CAA lors de l'émission ou du renouvellement d'un certificat ACM ?

Erreur lors de l'obtention de la validation du certificat DNS en attente pour un domaine personnalisé

- Vérifiez si vous avez ignoré une étape importante de la configuration du domaine personnalisé. Vérifiez également si vous avez mal configuré un enregistrement DNS à l'aide d'un outil de recherche DNS tel que DiG. Vérifiez en particulier les erreurs suivantes :
  - Toutes les étapes manquées.
  - Caractères non pris en charge tels que les guillemets doubles dans les enregistrements DNS.
- Corrigez les erreurs.
- Réessayez de lier le domaine personnalisé à App Runner.

Pour obtenir des instructions sur la résolution des erreurs de validation CAA, consultez ce qui suit.

- Validation du DNS
- the section called “noms de domaine personnalisés”

Commandes de dépannage de base

- Vérifiez qu'un service est disponible.

```
aws apprunner list-services
```

- Décrivez un service et vérifiez son état.

```
aws apprunner describe-service --service-arn
```

- Vérifiez le statut du domaine personnalisé.

```
aws apprunner describe-custom-domains --service-arn
```

- Répertoriez toutes les opérations en cours.

```
aws apprunner list-operations --service-arn
```

Renouvellement du certificat de domaine personnalisé

Lorsque vous ajoutez un domaine personnalisé à votre service, App Runner vous fournit un ensemble d'enregistrements CNAME que vous ajoutez à votre serveur DNS. Ces enregistrements CNAME incluent les enregistrements de certificats. App Runner utilise AWS Certificate Manager (ACM) pour vérifier le domaine. App Runner valide ces enregistrements DNS pour garantir le maintien de la propriété de ce domaine. Si vous supprimez les enregistrements CNAME de votre zone DNS, App Runner ne peut plus valider les enregistrements DNS et le certificat de domaine personnalisé ne se renouvelle pas automatiquement.

Cette section explique comment résoudre les problèmes de renouvellement de certificats de domaine personnalisés suivants :

- the section called “Le CNAME est supprimé du serveur DNS”.
- the section called “Le certificat a expiré”.

Le CNAME est supprimé du serveur DNS

- Récupérez vos enregistrements CNAME à l'aide de l'DescribeCustomDomainsAPI ou des paramètres de domaine personnalisés de la console App Runner. Pour plus d'informations sur CNAMEs le stockage, consultez CertificateValidationRecords.
- Ajoutez les enregistrements CNAME de validation du certificat à votre serveur DNS. App Runner peut ensuite valider que vous êtes propriétaire du domaine. Après avoir ajouté les enregistrements CNAME, la propagation des enregistrements DNS peut prendre jusqu'à 30 minutes. App Runner et ACM peuvent également mettre plusieurs heures à réessayer le processus de renouvellement

du certificat. Pour obtenir des instructions sur la façon d'ajouter des enregistrements CNAME, consultezthe section called “Gérez des domaines personnalisés”.

Le certificat a expiré

- Dissociez (dissociez) puis associez (liez) le domaine personnalisé de votre service App Runner à l'aide de la console ou de l'API App Runner. App Runner crée de nouveaux enregistrements CNAME de validation de certificat.
- Ajoutez les nouveaux enregistrements CNAME de validation de certificat à votre serveur DNS.

  Pour obtenir des instructions sur la façon de dissocier (dissocier) et d'associer (lier) le domaine personnalisé, consultez. the section called “Gérez des domaines personnalisés”

Comment puis-je vérifier que le certificat a été renouvelé avec succès ?

Vous pouvez vérifier le statut de vos enregistrements de certificats pour vérifier que votre certificat a été renouvelé avec succès. Vous pouvez vérifier l'état des certificats à l'aide d'outils tels que curl.

Pour plus d'informations sur le renouvellement des certificats, consultez les liens suivants :

- Pourquoi mon certificat ACM est-il marqué comme non éligible au renouvellement ?
- Renouvellement géré pour les certificats ACM
- Validation du DNS

Erreur de routage des demandes HTTP/HTTPS

Cette section explique comment résoudre les problèmes et les erreurs que vous pourriez rencontrer lors du routage du HTTP/HTTPS trafic vers les points de terminaison de votre service App Runner.

Erreur 404 Introuvable lors de l'envoi de HTTP/HTTPS trafic vers les points de terminaison du service App Runner

- Vérifiez que la `Host Header` requête HTTP pointe vers l'URL du service, car App Runner utilise les informations d'en-tête de l'hôte pour acheminer les demandes. La plupart des clients et cURL des navigateurs Web pointent automatiquement l'en-tête de l'hôte vers l'URL du service. Si votre client ne définit pas l'URL du service comme étant le`Host Header`, vous recevez un `404 Not Found` message d'erreur.

Example En-tête d'hôte incorrect

```
$ ~ curl -I -H "host: foobar.com" https://testservice.awsapprunner.com/
HTTP/1.1 404 Not Found
transfer-encoding: chunked
```

Example En-tête d'hôte correct

```
$ ~ curl -I -H "host: testservice.awsapprunner.com" https://
testservice.awsapprunner.com/
HTTP/1.1 200 OK
content-length: 11772
content-type: text/html; charset=utf-8
```

- Vérifiez que votre client définit correctement l'indicateur de nom de serveur (SNI) pour les demandes acheminées vers des services publics ou privés. Pour la terminaison du protocole TLS et le routage des demandes, App Runner utilise le SNI défini dans la connexion HTTPS.

Lorsque le service ne parvient pas à se connecter à Amazon RDS ou au service en aval

Votre service peut présenter un problème de configuration réseau s'il ne parvient pas à se connecter à une base de données Amazon RDS ou à une autre application ou service en aval. Cette rubrique décrit les étapes à suivre pour déterminer s'il existe des problèmes liés à la configuration de votre réseau et les options permettant de les corriger. Pour en savoir plus sur la configuration du trafic sortant pour App Runner, consultezActivation de l'accès VPC pour le trafic sortant.

> **Note**
>
> Pour afficher la configuration de votre connecteur VPC, dans le volet de navigation de gauche de la console App Runner, sélectionnez Configuration réseau. Sélectionnez ensuite l'onglet Trafic sortant. Sélectionnez un connecteur VPC. La page suivante affiche des informations sur le connecteur VPC. Sur cette page, vous pouvez consulter et analyser les éléments suivants : les sous-réseaux, les groupes de sécurité et les services App Runner qui utilisent le VPC.

Pour déterminer la cause de l'incapacité de votre application à se connecter à un autre service en aval

1. Assurez-vous que les sous-réseaux utilisés dans les connecteurs VPC sont des sous-réseaux privés. Si un connecteur est configuré avec un sous-réseau public, votre service rencontrera des erreurs, car l'hyperplan sous-jacent ENIs (interfaces réseau élastiques) de chaque sous-réseau ne possède pas d'espace IP public.

   Si vos connecteurs VPC utilisent des sous-réseaux publics, vous disposez des options suivantes pour corriger cette configuration :

   a. Créez un nouveau sous-réseau privé et utilisez-le à la place du sous-réseau public pour le connecteur VPC. Pour plus d'informations, consultez la section Sous-réseaux de votre VPC dans le guide de l'utilisateur Amazon VPC.
   b. Routez le sous-réseau public existant via des passerelles NAT. Pour plus d'informations, consultez la section Passerelles NAT dans le guide de l'utilisateur Amazon VPC.
2. Vérifiez que les règles d'entrée et de sortie du groupe de sécurité pour le connecteur VPC sont correctes. Dans le volet de navigation gauche de la console App Runner, sélectionnez Configuration réseau > Trafic sortant. Sélectionnez le connecteur VPC dans la liste. La page suivante répertorie les groupes de sécurité que vous pouvez sélectionner pour inspecter.
3. Vérifiez que les règles entrantes et sortantes du groupe de sécurité sont correctes pour l'instance RDS ou tout autre service en aval auquel vous tentez de vous connecter. Pour plus d'informations, consultez le guide de service du service en aval auquel votre application App Runner essaie de se connecter.
4. Pour vérifier qu'il n'existe aucun autre type de problème de configuration réseau en dehors de vos configurations App Runner, essayez de vous connecter au RDS ou au service en aval en dehors d'App Runner :

   a. À partir d'une instance Amazon EC2 dans le même VPC, essayez de vous connecter à l'instance ou au service RDS.
   b. Si vous essayez de vous connecter à un point de terminaison VPC de service, vérifiez la connectivité en accédant au même point de terminaison depuis une instance EC2 du même VPC.
5. Si l'un des tests de connexion de l'étape 4 échoue, il est fort probable qu'il y ait un problème en dehors de vos configurations App Runner avec une autre ressource de votre AWS compte. Contactez le AWS Support pour obtenir de l'aide afin d'isoler et de résoudre le problème lié à vos autres configurations réseau.

6. Si vous vous connectez avec succès à l'instance RDS ou au service en aval en suivant les instructions de l'étape 4, suivez les instructions de cette étape. Nous vérifierons si le trafic entre dans l'ENI en activant et en inspectant les journaux de flux Hyperplane ENI.

   > **Note**
   >
   > Pour pouvoir effectuer ces étapes et obtenir les informations du journal de flux ENI requises, la tentative de connexion au RDS ou au service en aval doit avoir lieu après le démarrage réussi de votre service App Runner. Votre application doit effectuer l'opération de connexion au service RDS ou au service en aval lorsqu'elle est en cours d'exécution. Sinon, ils ENIs pourraient être nettoyés dans le cadre des flux de travail de restauration d'App Runner. Cette approche garantit qu'ils ENIs restent disponibles pour une enquête plus approfondie.

   a. Depuis la AWS console, lancez la console EC2.

   b. Dans le volet de navigation de gauche, dans le groupe Réseau et sécurité, sélectionnez Interfaces réseau.

   c. Accédez aux colonnes Type d'interface et Description pour les localiser ENIs dans les sous-réseaux associés au connecteur VPC. Ils auront les modèles de dénomination suivants.

      - Type d'interface : fargate
      - Description : commence par AWSAppRunner ENI(exemple : AWSAppRunner ENI - abcde123-abcd-1234-1234-abcde1233456)

   d. Utilisez les cases à cocher situées au début des lignes pour sélectionner celles ENIs qui s'appliquent.

   e. Dans le menu Actions, sélectionnez Créer un journal de flux.

   f. Entrez les informations dans les instructions et sélectionnez Créer un flog de flux au bas de la page.

   g. Inspectez le journal de flux généré.

      - Si du trafic entrait dans l'ENI lorsque vous testiez la connexion, le problème n'est pas lié à la configuration de l'ENI. Des problèmes de configuration réseau peuvent survenir avec une autre ressource de votre AWS compte que les services App Runner. Contactez le AWS Support pour obtenir de l'aide supplémentaire.

- Si le trafic n'entrait pas dans l'ENI lorsque vous testiez la connexion, nous vous conseillons de contacter le AWS Support pour voir s'il existe des problèmes connus avec le service Fargate.

h. Utilisez l'outil Network Reachability Analyzer. Cet outil permet de déterminer les erreurs de configuration du réseau en identifiant les composants bloquants lorsqu'une source du chemin réseau virtuel n'est pas accessible. Pour plus d'informations, voir Qu'est-ce que Reachability Analyzer ? dans le guide Amazon VPC Reachability Analyzer.

   Entrez l'ENI App Runner comme source et le RDS ENI comme destination.

7. Si vous ne parvenez pas à préciser le problème, ou si vous ne parvenez toujours pas à vous connecter au service RDS ou au service en aval après avoir effectué les étapes précédentes, nous vous conseillons de contacter le AWS Support pour obtenir une assistance supplémentaire.

Lorsqu'il n'y a pas assez d'adresses IP pour le lancement d'instances ou le dimensionnement

> **Note**
>
> Pour les services publics, App Runner ne crée pas d'interface réseau élastique (ENI) dans votre VPCs ordinateur. Vos services publics ne sont donc pas affectés par cette modification.

Ce guide vous aide à résoudre les erreurs d'épuisement des adresses IP que vous pouvez rencontrer sur les services App Runner lorsque l'accès VPC pour le trafic sortant est activé.

App Runner lancera des instances dans les sous-réseaux associés à votre connecteur VPC. App Runner crée 1 ENI par instance dans le sous-réseau où votre instance est lancée. Chaque ENI utilise une adresse IP privée dans ce sous-réseau. Le nombre de sous-réseaux IPs disponibles est fixe, en fonction du bloc CIDR associé à ce sous-réseau. Si App Runner ne trouve pas de sous-réseau suffisant IPs pour créer un ENI, il ne pourra pas lancer de nouvelles instances pour votre service App Runner. Cela peut entraîner des problèmes lors de l'extension de vos services. Dans ce cas, vous verrez les journaux d'événements d'App Runner indiquant qu'App Runner est incapable de trouver les sous-réseaux disponibles IPs. Vous pouvez mettre à jour vos services en suivant les instructions ci-dessous pour résoudre ces erreurs.

Comment mettre à jour vos services pour en avoir davantage à votre disposition IPs

Le nombre d'adresses IP disponibles dans un sous-réseau dépend du bloc CIDR associé à ce sous-réseau. Les blocs CIDR associés à un sous-réseau ne peuvent pas être mis à jour après leur création. Les connecteurs VPC App Runner ne peuvent pas non plus être mis à jour une fois qu'ils ont été créés. Pour optimiser vos services App Runner en activant l'accès VPC pour le trafic sortant : IPs

1. Créez de nouveaux sous-réseaux avec un bloc CIDR plus grand.
2. Créez un nouveau connecteur VPC avec le ou les nouveaux sous-réseaux.
3. Mettez à jour votre service App Runner pour utiliser le nouveau connecteur VPC.

Calcul des IPs besoins pour vos services

Avant d'essayer de créer de nouveaux sous-réseaux avec des blocs CIDR plus grands, déterminez le nombre dont IPs vous aurez besoin pour tous vos services App Runner. Nous vous recommandons de calculer le nombre de pièces IPs nécessaires dans votre connecteur comme suit :

1. Pour chaque service dont l'accès VPC est activé pour le trafic sortant, notez la taille maximale (nombre maximum d'instances) dans la configuration de dimensionnement automatique.
2. Additionnez les valeurs de tous les services.
3. Doublez cette somme pour tenir compte des nouvelles instances lancées lors de déploiements bleu-vert.

Exemple

Supposons que deux services A et B utilisent le même connecteur VPC.

1. La taille maximale du service A est fixée à 25.
2. La taille maximale du service B est fixée à 15.

Obligatoire IPs = 2 × (25 + 15) = 80

Assurez-vous que vos sous-réseaux ont au moins 80 sous-réseaux disponibles IPs combinés.

Création de nouveaux sous-réseaux

1. Déterminez la taille de bloc CIDR nécessaire pour IPv4 utiliser cette formule (notez que 5 IPs sont réservés par AWS : Dimensionnement du sous-réseau)

   ```
   Number of available IP addresses = 2^(32 - prefix length) - 5
   ```

   ```
   Example :
   For 192.168.1.0/24:
   Prefix length is 24
   Number of available IP addresses = 2^(32 - 24) - 5 = 2^8-5 = 251 IP addresses

   For 10.0.0.0/16:
   Prefix length is 16
   Number of available IP addresses = 2^(32 - 16) - 5 = 2^16-5 = 65,531 IP addresses

   Quick reference:
   /24 = 251 IP addresses
   /16 = 65,531 IP addresses
   ```

2. Créez un nouveau sous-réseau à l'aide de l'AWS EC2 CLI.

   ```
   aws ec2 create-subnet --vpc-id <my-vpc-id> --cidr-block <cidr-block>
   ```

   Exemple (crée un sous-réseau avec 4 096) : IPs

   ```
   aws ec2 create-subnet --vpc-id my-vpc-id --cidr-block 10.0.0.0/20
   ```

3. Créez un nouveau connecteur VPC. Voir : Gérer l'accès aux VPC
4. Mettez à jour vos services avec le trafic sortant vers le VPC activé pour utiliser ce nouveau connecteur VPC. App Runner commencera à utiliser les nouveaux sous-réseaux une fois votre service mis à jour.

> **Note**
>
> VPCs sont également limités par le nombre de blocs CIDR disponibles IPs pouvant être alloués aux sous-réseaux. Si vous ne parvenez pas à créer des sous-réseaux avec des blocs

d'adresse CIDR plus grands, vous devrez peut-être mettre à jour votre VPC avec des blocs d'adresse CIDR secondaires avant de créer le ou les nouveaux sous-réseaux.

Joindre des blocs CIDR secondaires à votre VPC

Associez un bloc CIDR secondaire à ce VPC.

```
aws ec2 associate-vpc-cidr-block --vpc-id <my-vpc-id> --cidr-block <cidr-block>
```

Exemple :

```
aws ec2 associate-vpc-cidr-block --vpc-id my-vpc-id --cidr-block 10.1.0.0/16
```

Vérification

Une fois que vous avez mis à jour votre service. Vous pouvez utiliser les méthodes suivantes pour vérifier votre correctif

1. Surveillez les journaux des événements : surveillez les journaux des événements de votre service App Runner pour vérifier qu'aucune nouvelle erreur d'indisponibilité d'IP ou d'ENI n'apparaît
2. Vérifiez le dimensionnement du service :
    1. Augmentez complètement le service en modifiant le nombre minimum d'instances dans votre configuration de mise à l'échelle automatique
    2. Vérifiez que toutes les nouvelles instances sont lancées sans aucune erreur liée à l'adresse IP
    3. Surveillez plusieurs événements de dimensionnement pour garantir des performances constantes
3. Bannière de console : si vous utilisez la console de gestion AWS, vérifiez qu'App Runner n'affiche plus de bannière signalant une insuffisance IPs.
4. Utilisation du VPC et des adresses IP des sous-réseaux :
    1. Utilisez le tableau de bord VPC ou les commandes CLI pour vérifier l'utilisation des adresses IP dans vos nouveaux sous-réseaux.

2. Vérifiez qu'il reste une bonne marge de disponibilité IPs après le déploiement de votre service

Pièges courants

Lorsque vous abordez le problème de l'épuisement des adresses IP dans les services App Runner, soyez conscient des problèmes potentiels suivants :

1. Planification inadéquate des adresses IP : la sous-estimation des besoins futurs en matière de propriété intellectuelle peut entraîner des problèmes d'épuisement récurrents. Procédez à une planification approfondie des capacités, en tenant compte de la croissance potentielle des services et des scénarios d'utilisation maximale.
2. Oublier l'utilisation des adresses IP à l'échelle du VPC : n'oubliez pas que les autres services AWS au sein du même VPC consomment également des adresses IP. Tenez compte des exigences IP de tous les services lorsque vous planifiez les configurations de votre VPC et de vos sous-réseaux.
3. Négliger de mettre à jour les services : après avoir créé de nouveaux sous-réseaux ou connecteurs VPC, assurez-vous de mettre à jour vos services App Runner pour utiliser les nouvelles configurations. Dans le cas contraire, l'utilisation de la plage IP épuisée se poursuivra.
4. Mauvaise compréhension des chevauchements de blocs CIDR : Lorsque vous ajoutez des blocs d'adresse CIDR secondaires à un VPC, assurez-vous qu'ils ne se chevauchent pas avec les blocs existants. Le chevauchement des blocs CIDR peut entraîner des conflits de routage et une ambiguïté des adresses IP.
5. Dépassement des limites du VPC : sachez qu'un VPC peut comporter un maximum de 5 blocs CIDR (1 principal et 4 secondaires). Planifiez l'expansion de votre espace d'adressage IP en tenant compte de ces contraintes.
6. Ignorer la distribution des sous-réseaux AZ : lors de la création de nouveaux sous-réseaux, assurez-vous qu'ils sont répartis sur plusieurs zones de disponibilité pour garantir une haute disponibilité et une tolérance aux pannes.
7. Oublier les limites de l'ENI : n'oubliez pas que le nombre de ENIs caractères pouvant être attachés aux instances est limité. Vérifiez que les limites de votre compte AWS correspondent à l'utilisation prévue de l'interface réseau.

En étant conscient de ces écueils, vous pouvez gérer plus efficacement vos ressources VPC et éviter les problèmes d'épuisement des adresses IP dans vos services App Runner.

Ressources supplémentaires

1. Documentation AWS VPC
2. Comprendre les blocs CIDR
3. Connecteurs VPC App Runner

Glossaire

1. ENI : Elastic Network Interface, une interface réseau virtuelle dans AWS.
2. CIDR : Classless Inter-Domain Routing, méthode d'allocation d'adresses IP.
3. Connecteur VPC : ressource qui permet à App Runner de se connecter à votre VPC.

Sécurité dans App Runner

La sécurité du cloud AWS est la priorité absolue. En tant que AWS client, vous bénéficiez de centres de données et d'architectures réseau conçus pour répondre aux exigences des entreprises les plus sensibles en matière de sécurité.

La sécurité est une responsabilité partagée entre vous AWS et vous. Le modèle de responsabilité partagée décrit ceci comme la sécurité du cloud et la sécurité dans le cloud :

- Sécurité du cloud : AWS est chargée de protéger l'infrastructure qui exécute les AWS services dans le AWS Cloud. AWS vous fournit également des services que vous pouvez utiliser en toute sécurité. Des auditeurs tiers testent et vérifient régulièrement l'efficacité de notre sécurité dans le cadre des programmes de AWS conformité Programmes de de conformité. Pour en savoir plus sur les programmes de conformité qui s'appliquent à AWS App Runner, voir AWS Services concernés par programme de conformitéAWS .
- Sécurité dans le cloud — Votre responsabilité est déterminée par le AWS service 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 d'App Runner. Les rubriques suivantes expliquent comment configurer App Runner pour atteindre vos objectifs de sécurité et de conformité. Vous apprendrez également à utiliser d'autres AWS services qui vous aident à surveiller et à sécuriser vos ressources App Runner.

Rubriques

- Protection des données dans App Runner
- Gestion des identités et des accès pour App Runner
- Enregistrement et surveillance dans App Runner
- Validation de conformité pour App Runner
- Résilience dans App Runner
- Sécurité de l'infrastructure dans AWS App Runner
- Utilisation d'App Runner avec des points de terminaison VPC
- Analyse de configuration et de vulnérabilité dans App Runner
- Bonnes pratiques de sécurité pour App Runner

Protection des données dans App Runner

Le modèle de responsabilité AWS partagée de s'applique à la protection des données dans AWS App Runner. Comme décrit dans ce modèle, AWS est chargé de protéger l'infrastructure mondiale qui gère tous les AWS Cloud. La gestion du contrôle de votre contenu hébergé sur cette infrastructure relève de votre responsabilité. Vous êtes également responsable des tâches de configuration et de gestion de la sécurité des Services AWS que vous utilisez. Pour plus d'informations sur la confidentialité des données, consultez Questions fréquentes (FAQ) sur la confidentialité des données. Pour en savoir plus sur la protection des données en Europe, consultez le billet de blog Modèle de responsabilité partagée d'AWS et RGPD (Règlement général sur la protection des données) sur le Blog de sécuritéAWS .

À des fins de protection des données, nous vous recommandons de protéger les Compte AWS informations d'identification et de configurer les utilisateurs individuels avec AWS IAM Identity Center ou Gestion des identités et des accès AWS (IAM). Ainsi, chaque utilisateur se voit attribuer uniquement les autorisations nécessaires pour exécuter ses tâches. Nous vous recommandons également de sécuriser vos données comme indiqué ci-dessous :

- Utilisez l'authentification multifactorielle (MFA) avec chaque compte.
- SSL/TLS À utiliser pour communiquer avec AWS les ressources. Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Configurez l'API et la journalisation de l'activité des utilisateurs avec AWS CloudTrail. Pour plus d'informations sur l'utilisation des CloudTrail sentiers pour capturer AWS des activités, consultez la section Utilisation des CloudTrail sentiers dans le guide de AWS CloudTrail l'utilisateur.
- Utilisez des solutions de AWS chiffrement, ainsi que tous les contrôles de sécurité par défaut qu'ils contiennent Services AWS.
- Utilisez des services de sécurité gérés avancés tels qu'Amazon Macie, qui contribuent à la découverte et à la sécurisation des données sensibles stockées dans Amazon S3.
- Si vous avez besoin de modules cryptographiques validés par la norme FIPS 140-3 pour accéder AWS via une interface de ligne de commande ou une API, utilisez un point de terminaison FIPS. Pour plus d'informations sur les points de terminaison FIPS disponibles, consultez Norme FIPS (Federal Information Processing Standard) 140-3.

Nous vous recommandons fortement de ne jamais placer d'informations confidentielles ou sensibles, telles que les adresses e-mail de vos clients, dans des balises ou des champs de texte libre tels que le champ Nom. Cela inclut lorsque vous travaillez avec App Runner ou autre Services AWS à

l'aide de la console, de l'API ou AWS SDKs. AWS CLI Toutes les données que vous entrez dans des balises ou des champs de texte de forme libre utilisés pour les noms peuvent être utilisées à des fins de facturation ou dans les journaux de diagnostic. Si vous fournissez une adresse URL à un serveur externe, nous vous recommandons fortement de ne pas inclure d'informations d'identification dans l'adresse URL permettant de valider votre demande adressée à ce serveur.

Pour d'autres sujets relatifs à la sécurité d'App Runner, consultezSécurité.

Rubriques

- Protection des données à l'aide du chiffrement
- Confidentialité du trafic inter-réseaux

Protection des données à l'aide du chiffrement

AWS App Runner lit la source de votre application (image source ou code source) à partir d'un référentiel que vous spécifiez et la stocke pour le déployer sur votre service. Pour de plus amples informations, veuillez consulter Architecture et concepts.

La protection des données fait référence à la protection des données en transit (lorsqu'elles sont acheminées vers et depuis App Runner) et au repos (lorsqu'elles sont stockées dans AWS des centres de données).

Pour plus d'informations sur la protection des données, consultez the section called "Protection des données".

Pour d'autres sujets relatifs à la sécurité d'App Runner, consultezSécurité.

Chiffrement en transit

Vous pouvez protéger les données en transit de deux manières : crypter la connexion à l'aide du protocole TLS (Transport Layer Security) ou utiliser le chiffrement côté client (l'objet étant chiffré avant d'être envoyé). Les deux méthodes sont valides pour protéger les données de votre application. Pour sécuriser la connexion, chiffrez-la à l'aide du protocole TLS chaque fois que votre application, ses développeurs, ses administrateurs et ses utilisateurs finaux envoient ou reçoivent des objets. App Runner configure votre application pour qu'elle reçoive du trafic via le protocole TLS.

Le chiffrement côté client n'est pas une méthode valide pour protéger l'image source ou le code que vous fournissez à App Runner pour le déploiement. App Runner a besoin d'accéder à la source de

votre application, elle ne peut donc pas être chiffrée. Veillez donc à sécuriser la connexion entre votre environnement de développement ou de déploiement et App Runner.

Chiffrement au repos et gestion des clés

Pour protéger les données inactives de votre application, App Runner chiffre toutes les copies stockées de l'image source ou du bundle de sources de votre application. Lorsque vous créez un service App Runner, vous pouvez fournir un AWS KMS key. Si vous en fournissez une, App Runner utilise la clé que vous avez fournie pour chiffrer votre source. Si vous n'en fournissez pas, App Runner en utilise un à la Clé gérée par AWS place.

Pour plus de détails sur les paramètres de création du service App Runner, consultez CreateService. Pour plus d'informations sur AWS Key Management Service (AWS KMS), consultez le manuel du AWS Key Management Service développeur.

Confidentialité du trafic inter-réseaux

App Runner utilise Amazon Virtual Private Cloud (Amazon VPC) pour créer des limites entre les ressources de votre application App Runner et contrôler le trafic entre celles-ci, votre réseau local et Internet. Pour plus d'informations sur la sécurité d'Amazon VPC, consultez la section Confidentialité du trafic interréseau dans Amazon VPC dans le guide de l'utilisateur Amazon VPC.

Pour plus d'informations sur l'association de votre application App Runner à un Amazon VPC personnalisé, consultez. the section called “Trafic sortant”

Pour plus d'informations sur la sécurisation des demandes adressées à App Runner à l'aide d'un point de terminaison VPC, consultez. the section called “Points de terminaison d’un VPC”

Pour plus d’informations sur la protection des données, consultez the section called “Protection des données”.

Pour d'autres sujets relatifs à la sécurité d'App Runner, consultezSécurité.

Gestion des identités et des accès pour App Runner

Gestion des identités et des accès AWS (IAM) est un outil Service AWS qui permet à un administrateur de contrôler en toute sécurité l'accès aux AWS ressources. Les administrateurs IAM contrôlent qui peut être authentifié (connecté) et autorisé (autorisé) à utiliser les ressources App Runner. IAM est un Service AWS outil que vous pouvez utiliser sans frais supplémentaires.

Pour d'autres sujets relatifs à la sécurité d'App Runner, consultezSécurité.

Rubriques

- Public ciblé
- Authentification par des identités
- Gestion de l’accès à l’aide de politiques
- Comment App Runner fonctionne avec IAM
- Exemples de politiques basées sur l'identité d'App Runner
- Utilisation de rôles liés à un service pour App Runner
- AWS politiques gérées pour AWS App Runner
- Résolution des problèmes d'identité et d'accès à App Runner

Public ciblé

La façon dont vous utilisez Gestion des identités et des accès AWS (IAM) varie en fonction de votre rôle :

- Utilisateur du service : demandez des autorisations à votre administrateur si vous ne pouvez pas accéder aux fonctionnalités (voir Résolution des problèmes d'identité et d'accès à App Runner)
- Administrateur du service : déterminez l’accès des utilisateurs et soumettez les demandes d’autorisation (voir Comment App Runner fonctionne avec IAM)
- Administrateur IAM : rédigez des politiques pour gérer l’accès (voir Exemples de politiques basées sur l'identité d'App Runner)

Authentification par des identités

L'authentification est la façon dont vous vous connectez à AWS l'aide de vos informations d'identification. Vous devez être authentifié en tant qu'utilisateur IAM ou en assumant un rôle IAM. Utilisateur racine d'un compte AWS

Vous pouvez vous connecter en tant qu'identité fédérée à l'aide d'informations d'identification provenant d'une source d'identité telle que AWS IAM Identity Center (IAM Identity Center), d'une authentification unique ou d'informations d'identification. Google/Facebook Pour plus d’informations sur la connexion, consultez Connexion à votre Compte AWS dans le Guide de l’utilisateur Connexion à AWS .

Pour l'accès par programmation, AWS fournit un SDK et une CLI pour signer les demandes de manière cryptographique. Pour plus d'informations, consultez Signature AWS Version 4 pour les demandes d'API dans le Guide de l'utilisateur IAM.

Compte AWS utilisateur root

Lorsque vous créez un Compte AWS, vous commencez par une seule identité de connexion appelée utilisateur Compte AWS root qui dispose d'un accès complet à toutes Services AWS les ressources. Il est vivement déconseillé d'utiliser l'utilisateur racine pour vos tâches quotidiennes. Pour les tâches qui requièrent des informations d'identification de l'utilisateur racine, consultez Tâches qui requièrent les informations d'identification de l'utilisateur racine dans le Guide de l'utilisateur IAM.

Utilisateurs et groupes IAM

Un utilisateur IAM est une identité qui dispose d'autorisations spécifiques pour une seule personne ou application. Nous vous recommandons d'utiliser ces informations d'identification temporaires au lieu des utilisateurs IAM avec des informations d'identification à long terme. Pour plus d'informations, voir Exiger des utilisateurs humains qu'ils utilisent la fédération avec un fournisseur d'identité pour accéder à AWS l'aide d'informations d'identification temporaires dans le guide de l'utilisateur IAM.

Les groupes IAM spécifient une collection d'utilisateurs IAM et permettent de gérer plus facilement les autorisations pour de grands ensembles d'utilisateurs. Pour plus d'informations, consultez Cas d'utilisation pour les utilisateurs IAM dans le Guide de l'utilisateur IAM.

Rôles IAM

Un rôle IAM est une identité dotée d'autorisations spécifiques qui fournit des informations d'identification temporaires. Vous pouvez assumer un rôle en passant d'un rôle d'utilisateur à un rôle IAM (console) ou en appelant une opération d' AWS API AWS CLI ou d'API. Pour plus d'informations, consultez Méthodes pour endosser un rôle dans le Guide de l'utilisateur IAM.

Les rôles IAM sont utiles pour l'accès des utilisateurs fédérés, les autorisations temporaires des utilisateurs IAM, les accès intercompte, les accès entre services et les applications exécutées sur Amazon EC2. Pour plus d'informations, consultez Accès intercompte aux ressources dans IAM dans le Guide de l'utilisateur IAM.

Gestion de l'accès à l'aide de politiques

Vous contrôlez l'accès en AWS créant des politiques et en les associant à AWS des identités ou à des ressources. Une politique définit les autorisations lorsqu'elles sont associées à une identité

ou à une ressource. AWS évalue ces politiques lorsqu'un directeur fait une demande. La plupart des politiques sont stockées AWS sous forme de documents JSON. Pour plus d'informations les documents de politique JSON, consultez Vue d'ensemble des politiques JSON dans le Guide de l'utilisateur IAM.

À l'aide de politiques, les administrateurs précisent qui a accès à quoi en définissant 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. Un administrateur IAM crée des politiques IAM et les ajoute aux rôles, que les utilisateurs peuvent ensuite assumer. Les politiques IAM définissent les autorisations quelle que soit la méthode que vous utilisez pour exécuter l'opération.

Politiques basées sur l'identité

Les stratégies basées sur l'identité sont des documents de stratégie d'autorisations JSON que vous attachez à une identité (utilisateur, groupe ou rôle). Ces politiques contrôlent les actions que peuvent exécuter ces identités, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l'identité, consultez Définition d'autorisations IAM personnalisées avec des politiques gérées par le client dans le Guide de l'utilisateur IAM.

Les politiques basées sur l'identité peuvent être des politiques intégrées (intégrées directement dans une seule identité) ou des politiques gérées (politiques autonomes associées à plusieurs identités). Pour découvrir comment choisir entre des politiques gérées et 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. Les exemples incluent les politiques de confiance de rôle IAM et les stratégies 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. Vous devez spécifier un principal dans une politique basée sur les ressources.

Les politiques basées sur les ressources sont des politiques en ligne situées dans ce service. Vous ne pouvez pas utiliser les politiques AWS gérées par IAM dans une stratégie basée sur les ressources.

Listes de contrôle d'accès (ACLs)

Les listes de contrôle d'accès (ACLs) contrôlent les principaux (membres du compte, utilisateurs ou rôles) autorisés à accéder à une ressource. ACLs sont similaires aux politiques basées sur les ressources, bien qu'elles n'utilisent pas le format de document de politique JSON.

Amazon S3 et AWS WAF Amazon VPC sont des exemples de services compatibles. ACLs Pour en savoir plus ACLs, consultez la présentation de la liste de contrôle d'accès (ACL) dans le guide du développeur Amazon Simple Storage Service.

Autres types de politique

AWS prend en charge des types de politiques supplémentaires qui peuvent définir les autorisations maximales accordées par les types de politiques les plus courants :

- Limites d'autorisations : une limite des autorisations définit le nombre maximum d'autorisations qu'une politique basée sur l'identité peut accorder à une entité IAM. Pour plus d'informations, consultez Limites d'autorisations pour des entités IAM dans le Guide de l'utilisateur IAM.
- Politiques de contrôle des services (SCPs) — Spécifiez les autorisations maximales pour une organisation ou une unité organisationnelle dans AWS Organizations. Pour plus d'informations, consultez Politiques de contrôle de service dans le Guide de l'utilisateur AWS Organizations .
- Politiques de contrôle des ressources (RCPs) : définissez le maximum d'autorisations disponibles pour les ressources de vos comptes. Pour plus d'informations, voir Politiques de contrôle des ressources (RCPs) dans le guide de AWS Organizations l'utilisateur.
- Politiques de session : politiques avancées que vous passez en tant que paramètre lorsque vous créez par programmation une session temporaire pour un rôle ou un utilisateur fédéré. Pour plus d'informations, consultez Politiques de session dans le Guide de l'utilisateur IAM.

Plusieurs types de politique

Lorsque plusieurs types de politiques s'appliquent à la requête, les autorisations en résultant sont plus compliquées à comprendre. Pour savoir comment AWS déterminer s'il faut autoriser une demande lorsque plusieurs types de politiques sont impliqués, consultez la section Logique d'évaluation des politiques dans le guide de l'utilisateur IAM.

Comment App Runner fonctionne avec IAM

Avant d'utiliser IAM pour gérer l'accès à AWS App Runner, vous devez connaître les fonctionnalités IAM disponibles avec App Runner. Pour obtenir une vue d'ensemble de la façon dont App Runner et les autres AWS services fonctionnent avec IAM, consultez la section AWS Services qui fonctionnent avec IAM dans le guide de l'utilisateur d'IAM.

Pour d'autres sujets relatifs à la sécurité d'App Runner, consultezSécurité.

Rubriques

- Politiques basées sur l'identité d'App Runner
- Politiques basées sur les ressources d'App Runner
- Autorisation basée sur les tags App Runner
- Autorisations utilisateur d'App Runner
- Rôles IAM dans App Runner

Politiques basées sur l'identité d'App Runner

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. App Runner prend en charge des actions, des ressources et des clés de condition spécifiques. Pour en savoir plus sur 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.

Actions

Les administrateurs peuvent utiliser les politiques AWS JSON 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. Intégration d'actions dans une politique afin d'accorder l'autorisation d'exécuter les opérations associées.

Les actions de politique dans App Runner utilisent le préfixe suivant avant l'action :`apprunner:`. Par exemple, pour accorder à une personne l'autorisation d'exécuter une instance Amazon EC2 avec l'opération d'API `RunInstances` Amazon EC2, vous incluez l'action `ec2:RunInstances` dans sa politique. Les déclarations de politique doivent inclure un élément `Action` ou `NotAction`. App

Runner définit son propre ensemble d'actions décrivant les tâches que vous pouvez effectuer avec ce service.

Pour spécifier plusieurs actions dans une seule déclaration, séparez-les par des virgules comme suit :

```
"Action": [
   "apprunner:CreateService",
   "apprunner:CreateConnection"
]
```

Vous pouvez aussi spécifier plusieurs actions à l'aide de caractères génériques (*). Par exemple, pour spécifier toutes les actions qui commencent par le mot `Describe`, incluez l'action suivante :

```
"Action": "apprunner:Describe*"
```

Pour consulter la liste des actions d'App Runner, consultez la section Actions définies par AWS App Runner dans la référence d'autorisation de service.

Ressources

Les administrateurs peuvent utiliser les politiques AWS JSON 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. Il est recommandé de définir une ressource à l'aide de son Amazon Resource Name (ARN). Pour les actions qui ne sont pas compatibles avec les autorisations de niveau ressource, utilisez un caractère générique (*) afin d'indiquer que l'instruction s'applique à toutes les ressources.

```
"Resource": "*"
```

Les ressources App Runner ont la structure ARN suivante :

```
arn:aws:apprunner:region:account-id:resource-type/resource-name[/resource-id]
```

Pour plus d'informations sur le format de ARNs, consultez Amazon Resource Names (ARNs) et AWS Service Namespaces dans le Références générales AWS.

Par exemple, pour spécifier le `my-service` service dans votre relevé, utilisez l'ARN suivant :

```
"Resource": "arn:aws:apprunner:us-east-1:123456789012:service/my-service"
```

Pour spécifier tous les services appartenant à un compte spécifique, utilisez le caractère générique (*) :

```
"Resource": "arn:aws:apprunner:us-east-1:123456789012:service/*"
```

Certaines actions d'App Runner, telles que celles relatives à la création de ressources, ne peuvent pas être effectuées sur une ressource spécifique. Dans ces cas-là, vous devez utiliser le caractère générique (*).

```
"Resource": "*"
```

Pour consulter la liste des types de ressources App Runner et leurs caractéristiques ARNs, consultez la section Ressources définies par AWS App Runner dans la référence d'autorisation de service. Pour savoir grâce à quelles actions vous pouvez spécifier l'ARN de chaque ressource, consultez Actions définies par AWS App Runner.

Clés de condition

Les administrateurs peuvent utiliser les politiques AWS JSON 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` indique à quel moment les instructions s'exécutent en fonction de critères définis. 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. Pour voir toutes les clés de condition AWS globales, voir les clés de contexte de condition AWS globales dans le guide de l'utilisateur IAM.

App Runner prend en charge l'utilisation de certaines clés de condition globales. Pour voir toutes les clés de condition AWS globales, consultez la section Clés contextuelles de condition AWS globale dans le guide de l'utilisateur IAM.

App Runner définit un ensemble de clés de condition spécifiques au service. En outre, App Runner prend en charge le contrôle d'accès basé sur des balises, qui est mis en œuvre à l'aide de clés de condition. Pour en savoir plus, consultez the section called “Autorisation basée sur les tags App Runner”.

Pour consulter la liste des clés de condition d'App Runner, consultez la section Clés de condition pour AWS App Runner la référence d'autorisation de service. Pour savoir avec quelles actions et ressources vous pouvez utiliser une clé de condition, consultez la section Actions définies par AWS App Runner.

Exemples

Pour consulter des exemples de politiques basées sur l'identité d'App Runner, consultez. Exemples de politiques basées sur l'identité d'App Runner

Politiques basées sur les ressources d'App Runner

App Runner ne prend pas en charge les politiques basées sur les ressources.

Autorisation basée sur les tags App Runner

Vous pouvez associer des tags aux ressources App Runner ou transmettre des tags dans une demande à App Runner. Pour contrôler l'accès basé sur des étiquettes, vous devez fournir les informations d'étiquette dans l'élément de condition d'une politique utilisant les clés de condition `apprunner:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* ou `aws:TagKeys`. Pour plus d'informations sur le balisage des ressources App Runner, consultezthe section called "Configuration".

Pour visualiser un exemple de politique basée sur l'identité permettant de limiter l'accès à une ressource en fonction des balises de cette ressource, consultez Contrôle de l'accès aux services App Runner en fonction des balises.

Autorisations utilisateur d'App Runner

Pour utiliser App Runner, les utilisateurs d'IAM doivent être autorisés à effectuer des actions App Runner. Une méthode courante pour accorder des autorisations aux utilisateurs consiste à associer une politique aux utilisateurs ou aux groupes IAM. Pour plus d'informations sur la gestion des autorisations utilisateur, consultez la section Modification des autorisations d'un utilisateur IAM dans le guide de l'utilisateur IAM.

App Runner fournit deux politiques gérées que vous pouvez associer à vos utilisateurs.

- `AWSAppRunnerReadOnlyAccess`— Accorde l'autorisation de répertorier et d'afficher les informations relatives aux ressources App Runner.
- `AWSAppRunnerFullAccess`— Accorde des autorisations pour toutes les actions d'App Runner.

Pour un contrôle plus précis des autorisations des utilisateurs, vous pouvez créer une politique personnalisée et l'associer à vos utilisateurs. Pour plus de détails, consultez la section Création de politiques IAM dans le guide de l'utilisateur IAM.

Pour des exemples de politiques utilisateur, voirthe section called “Stratégies utilisateur”.

Rôles IAM dans App Runner

Un rôle IAM est une entité au sein de vous Compte AWS qui possède des autorisations spécifiques.

Rôles liés à un service

Les rôles liés aux AWS services permettent aux services d'accéder aux ressources d'autres services pour effectuer une action en votre nom. Les rôles liés à un service s’affichent dans votre compte IAM et sont la propriété du service. Un administrateur IAM peut consulter, mais ne peut pas modifier, les autorisations concernant les rôles liés à un service.

App Runner prend en charge les rôles liés aux services. Pour plus d'informations sur la création ou la gestion des rôles liés à un service App Runner, consultez. the section called “Utilisation des rôles liés à un service”

Rôles du service

Cette fonction permet à un service d’endosser une fonction du service en votre nom. Ce rôle autorise le service à accéder à des ressources d’autres services pour effectuer une action en votre nom. Les rôles de service s’affichent dans votre compte IAM et sont la propriété du compte. Cela signifie qu'un utilisateur IAM peut modifier les autorisations associées à ce rôle. Toutefois, une telle action peut perturber le bon fonctionnement du service.

App Runner prend en charge quelques rôles de service.

Rôle d'accès

Le rôle d'accès est un rôle qu'App Runner utilise pour accéder aux images de votre compte Amazon Elastic Container Registry (Amazon ECR). Il est obligatoire pour accéder à une image dans Amazon ECR, mais pas dans Amazon ECR Public.

Avant de créer un service basé sur une image dans Amazon ECR, utilisez IAM pour créer un rôle de service. Utilisez la politique gérée `AWSAppRunnerServicePolicyForECRAccess`dans votre rôle de service. Vous pouvez ensuite transmettre ce rôle à App Runner lorsque vous appelez

l'CreateServiceAPI dans le AuthenticationConfigurationmembre du SourceConfigurationparamètre ou lorsque vous utilisez la console App Runner pour créer un service.

> **Note**
>
> Si vous créez votre propre politique personnalisée pour votre rôle d'accès, veillez à spécifier l'`"Resource": "*"` `ecr:GetAuthorizationToken`action. Les jetons peuvent être utilisés pour accéder à n'importe quel registre Amazon ECR auquel vous avez accès.

Lorsque vous créez votre rôle d'accès, veillez à ajouter une politique de confiance qui déclare le principal de service App Runner `build.apprunner.amazonaws.com` comme une entité de confiance.

Politique de confiance pour un rôle d'accès

JSON

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

Si vous utilisez la console App Runner pour créer un service, la console peut automatiquement créer un rôle d'accès pour vous et le choisir pour le nouveau service. La console répertorie également les autres rôles de votre compte, et vous pouvez en sélectionner un autre si vous le souhaitez.

Rôle d’instance

Le rôle d'instance est un rôle facultatif qu'App Runner utilise pour fournir des autorisations aux actions de AWS service dont les instances de calcul de votre service ont besoin. Vous devez fournir

un rôle d'instance à App Runner si le code de votre application appelle AWS actions (APIs). Intégrez les autorisations requises dans votre rôle d'instance ou créez votre propre politique personnalisée et utilisez-la dans le rôle d'instance. Nous n'avons aucun moyen d'anticiper les appels utilisés par votre code. Par conséquent, nous ne fournissons pas de politique gérée à cette fin.

Avant de créer un service App Runner, utilisez IAM pour créer un rôle de service avec les politiques personnalisées ou intégrées requises. Vous pouvez ensuite transmettre ce rôle à App Runner en tant que rôle d'instance lorsque vous appelez l'CreateServiceAPI dans le `InstanceRoleArn` membre du InstanceConfigurationparamètre ou lorsque vous utilisez la console App Runner pour créer un service.

Lorsque vous créez votre rôle d'instance, veillez à ajouter une politique de confiance qui déclare le principal de service App Runner `tasks.apprunner.amazonaws.com` comme une entité de confiance.

Politique de confiance pour un rôle d'instance

JSON

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

Si vous utilisez la console App Runner pour créer un service, la console répertorie les rôles de votre compte et vous pouvez sélectionner le rôle que vous avez créé à cette fin.

Pour plus d'informations sur la création d'un service, consultezthe section called “Création”.

Exemples de politiques basées sur l'identité d'App Runner

Par défaut, les utilisateurs et les rôles IAM ne sont pas autorisés à créer ou à modifier AWS App Runner des ressources. Ils ne peuvent pas non plus effectuer de tâches à l'aide de l' AWS API AWS Management Console AWS CLI, ou. Un administrateur IAM doit créer des politiques IAM autorisant les utilisateurs et les rôles à exécuter des opérations d'API spécifiques sur les ressources spécifiées dont ils ont besoin. Il doit ensuite attacher ces politiques aux utilisateurs ou aux groupes IAM ayant besoin de ces autorisations.

Pour savoir comment créer une stratégie IAM basée sur l'identité à l'aide de ces exemples de documents de stratégie JSON, veuillez consulter Création de stratégies dans l'onglet JSON dans le Guide de l'utilisateur IAM.

Pour d'autres sujets relatifs à la sécurité d'App Runner, consultezSécurité.

Rubriques

- Bonnes pratiques en matière de politiques
- Stratégies utilisateur
- Contrôle de l'accès aux services App Runner en fonction des balises

Bonnes pratiques en matière de politiques

Les politiques basées sur l'identité déterminent si quelqu'un peut créer, accéder ou supprimer des ressources App Runner 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 :

- Commencez AWS par les politiques gérées et passez aux autorisations du moindre privilège : pour commencer à accorder des autorisations à vos utilisateurs et à vos charges de travail, utilisez les politiques AWS gérées qui accordent des autorisations pour de nombreux cas d'utilisation courants. Ils sont disponibles dans votre Compte AWS. Nous vous recommandons de réduire davantage les autorisations en définissant des politiques gérées par les AWS clients spécifiques à vos cas d'utilisation. Pour plus d’informations, consultez politiques gérées par AWS ou politiques gérées par AWS pour les activités professionnelles dans le Guide de l’utilisateur IAM.
- Accordez 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.

- Utilisez 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 par le biais d'un service spécifique Service AWS, tel que CloudFormation. Pour plus d'informations, consultez Conditions pour éléments de politique JSON IAM dans le Guide de l'utilisateur IAM.
- Utilisez l'Analyseur d'accès IAM pour valider vos politiques IAM afin de garantir des autorisations sécurisées et fonctionnelles : l'Analyseur d'accès IAM 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 politiques avec IAM Access Analyzer dans le Guide de l'utilisateur IAM.
- Exiger l'authentification multifactorielle (MFA) : si vous avez un scénario qui nécessite des utilisateurs IAM ou un utilisateur root, activez l'authentification MFA pour une sécurité accrue. Compte AWS Pour exiger la MFA lorsque des opérations d'API sont appelées, ajoutez des conditions MFA à vos politiques. Pour plus d'informations, consultez Sécurisation de l'accès aux API avec 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.

Stratégies utilisateur

Pour accéder à la console App Runner, les utilisateurs IAM doivent disposer d'un ensemble minimal d'autorisations. Ces autorisations doivent vous permettre de répertorier et d'afficher les détails des ressources App Runner de votre Compte AWS. Si vous créez une politique basée sur l'identité qui est plus restrictive que les autorisations minimales requises, la console ne fonctionnera pas comme prévu pour les utilisateurs dotés de cette politique.

App Runner fournit deux politiques gérées que vous pouvez associer à vos utilisateurs.

- `AWSAppRunnerReadOnlyAccess`— Accorde l'autorisation de répertorier et d'afficher les informations relatives aux ressources App Runner.
- `AWSAppRunnerFullAccess`— Accorde des autorisations pour toutes les actions d'App Runner.

Pour garantir que les utilisateurs peuvent utiliser la console App Runner, associez au minimum la politique `AWSAppRunnerReadOnlyAccess` gérée aux utilisateurs. Vous pouvez joindre la politique `AWSAppRunnerFullAccess` gérée à la place, ou ajouter des autorisations supplémentaires spécifiques, pour permettre aux utilisateurs de créer, de modifier et de supprimer des ressources. Pour plus d'informations, consultez Ajout d'autorisations à un utilisateur dans le Guide de l'utilisateur IAM.

Il n'est pas nécessaire d'accorder des autorisations de console minimales aux utilisateurs qui appellent uniquement l'API AWS CLI ou l' AWS API. Au lieu de cela, autorisez uniquement l'accès aux actions correspondant à l'opération d'API que vous souhaitez autoriser les utilisateurs à effectuer.

Les exemples suivants illustrent les politiques utilisateur personnalisées. Vous pouvez les utiliser comme points de départ pour définir vos propres politiques utilisateur personnalisées. Copiez l'exemple ou supprimez des actions, limitez les ressources et ajoutez des conditions.

Exemple : politique utilisateur de gestion de la console et des connexions

Cet exemple de politique permet l'accès à la console ainsi que la création et la gestion des connexions. Il n'autorise pas la création et la gestion du service App Runner. Il peut être associé à un utilisateur dont le rôle est de gérer l'accès du service App Runner aux ressources du code source.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "apprunner:List*",
        "apprunner:Describe*",
        "apprunner:CreateConnection",
        "apprunner:DeleteConnection"
      ],
      "Resource": "*"
```

```
    }
  ]
}
```

Exemple : politiques utilisateur utilisant des clés de condition

Les exemples de cette section illustrent les autorisations conditionnelles qui dépendent de certaines propriétés de ressources ou de certains paramètres d'action.

Cet exemple de politique permet de créer un service App Runner mais interdit d'utiliser une connexion nomméeprod.

JSON

```
  { "Version":"2012-10-17",
  "Statement":
     [ { "Sid": "AllowCreateAppRunnerServiceWithNonProdConnections",
         "Effect": "Allow",
         "Action": "apprunner:CreateService",
         "Resource": "*",
         "Condition":
            { "ArnNotLike":
               {"apprunner:ConnectionArn":"arn:aws:apprunner:*:*:connection/prod/
*"}
            }
       }
     ]
   }
```

Cet exemple de politique permet de mettre à jour un service App Runner nommé preprod uniquement avec une configuration de dimensionnement automatique nomméepreprod.

JSON

```
{
    "Version":"2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "AllowUpdatePreProdAppRunnerServiceWithPreProdASConfig",
            "Effect": "Allow",
            "Action": "apprunner:UpdateService",
            "Resource": "arn:aws:apprunner:*:*:service/preprod/*",
            "Condition": {
                "ArnLike": {
                    "apprunner:AutoScalingConfigurationArn":
 "arn:aws:apprunner:us-east-1:*:autoscalingconfiguration/preprod/*"
                }
            }
        }
    ]
}
```

Contrôle de l'accès aux services App Runner en fonction des balises

Vous pouvez utiliser des conditions dans votre politique basée sur l'identité pour contrôler l'accès aux ressources App Runner en fonction de balises. Cet exemple montre comment créer une politique permettant de supprimer un service App Runner. Toutefois, l'autorisation est accordée uniquement si l'étiquette de service `Owner` a la valeur du nom d'utilisateur de cet utilisateur. Cette politique accorde également les autorisations nécessaires pour réaliser cette action sur la console.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "ListServicesInConsole",
      "Effect": "Allow",
      "Action": "apprunner:ListServices",
      "Resource": "*"
    },
    {
      "Sid": "DeleteServiceIfOwner",
      "Effect": "Allow",
      "Action": "apprunner:DeleteService",
      "Resource": "arn:aws:apprunner:us-east-1:*:service/*",
      "Condition": {
```

```
        "StringEquals": {"aws:ResourceTag/Owner": "${aws:username}"}
      }
    }
  ]
}
```

Vous pouvez rattacher cette politique aux utilisateurs IAM de votre compte. Si un utilisateur nommé `richard-roe` tente de supprimer un service App Runner, le service doit être étiqueté `Owner=richard-roe` ou`owner=richard-roe`. Dans le cas contraire, l'utilisateur se voit refuser l'accès. La clé de condition d'étiquette `Owner` correspond à la fois à `Owner` et à `owner`, car les noms de clé de condition ne sont pas sensibles à la casse. Pour plus d'informations, veuillez consulter la rubrique Éléments de stratégie JSON IAM : Condition dans le Guide de l'utilisateur IAM.

Utilisation de rôles liés à un service pour App Runner

AWS App Runner utilise des Gestion des identités et des accès AWS rôles liés à un service (IAM). Un rôle lié à un service est un type unique de rôle IAM directement lié à App Runner. Les rôles liés à un service sont prédéfinis par App Runner et incluent toutes les autorisations dont le service a besoin pour appeler d'autres AWS services en votre nom.

Rubriques

- Utilisation des rôles pour la gestion
- Utilisation des rôles pour la mise en réseau

Utilisation des rôles pour la gestion

AWS App Runner utilise des Gestion des identités et des accès AWS rôles liés à un service (IAM). Un rôle lié à un service est un type unique de rôle IAM directement lié à App Runner. Les rôles liés à un service sont prédéfinis par App Runner et incluent toutes les autorisations dont le service a besoin pour appeler d'autres AWS services en votre nom.

Un rôle lié à un service facilite la configuration d'App Runner, car vous n'avez pas à ajouter manuellement les autorisations nécessaires. App Runner définit les autorisations associées à ses rôles liés aux services et, sauf indication contraire, seul App Runner peut assumer ses rôles. Les autorisations définies comprennent la politique de confiance 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. Cela protège vos ressources App Runner, car vous ne pouvez pas supprimer par inadvertance l'autorisation d'accès aux ressources.

Pour plus d'informations sur les autres services qui prennent en charge les rôles liés à un service, consultez Services AWS qui fonctionnent avec IAM et recherchez les services avec un Oui dans la colonne Rôle lié à un service. Choisissez un Oui ayant un lien permettant de consulter les détails du rôle pour ce service.

Autorisations de rôle liées au service pour App Runner

App Runner utilise le rôle lié au service nommé. AWSServiceRoleForAppRunner

Le rôle permet à App Runner d'effectuer les tâches suivantes :

- Transférez les journaux vers les groupes de CloudWatch journaux Amazon Logs.
- Créez des règles Amazon CloudWatch Events pour vous abonner aux push d'images Amazon Elastic Container Registry (Amazon ECR).
- Envoyez les informations de suivi à AWS X-Ray.

Le rôle AWSService RoleForAppRunner lié à un service fait confiance aux services suivants pour assumer le rôle :

- `apprunner.amazonaws.com`

Les politiques d'autorisation du rôle AWSService RoleForAppRunner lié au service contiennent toutes les autorisations dont App Runner a besoin pour effectuer des actions en votre nom :

- Politique gérée `AppRunnerServiceRolePolicy`
- Politique en matière de traçage par rayons X — Consultez le contenu de la politique suivant.

Politique en matière de traçage par rayons X

JSON

```
{
  "Version":"2012-10-17",
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "xray:PutTraceSegments",
        "xray:PutTelemetryRecords",
        "xray:GetSamplingRules",
        "xray:GetSamplingTargets",
        "xray:GetSamplingStatisticSummaries"
      ],
      "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 en savoir plus, consultez Service-Linked Role Permissions (autorisations du rôle lié à un service) dans le Guide de l'utilisateur IAM.

Création d'un rôle lié à un service pour App Runner

Vous n'avez pas besoin de créer manuellement un rôle lié à un service. Lorsque vous créez un service App Runner dans AWS Management Console, l' AWS CLI API ou l' AWS API, App Runner crée 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 service App Runner, App Runner crée à nouveau le rôle lié au service pour vous.

Modification d'un rôle lié à un service pour App Runner

App Runner ne vous permet pas de modifier le rôle AWSService RoleForAppRunner lié au service. Après avoir 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.

Supprimer un rôle lié à un service pour App Runner

Si vous n’avez plus besoin d’utiliser une fonctionnalité 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. Cependant, vous devez nettoyer votre rôle lié à un service avant de pouvoir le supprimer manuellement.

Nettoyage d’un rôle lié à un service

Avant de pouvoir utiliser IAM pour supprimer un rôle lié à un service, vous devez supprimer toutes les ressources utilisées par le rôle.

Dans App Runner, cela signifie supprimer tous les services App Runner de votre compte. Pour en savoir plus sur la suppression des services App Runner, consultezthe section called “Suppression”.

> **Note**
>
> Si le service App Runner utilise le rôle lorsque vous essayez de supprimer les ressources, la suppression risque d'échouer. Si cela se produit, patientez quelques minutes et réessayez.

Suppression manuelle du rôle lié au service

Utilisez la console IAM, le AWS CLI, ou l' AWS API pour supprimer le rôle lié au AWSService RoleForAppRunner service. Pour en savoir plus, consultez Suppression d’un rôle lié à un service dans le Guide de l’utilisateur IAM.

Régions prises en charge pour les rôles liés au service App Runner

App Runner prend en charge l'utilisation de rôles liés à un service dans toutes les régions où le service est disponible. Pour plus d’informations, consultez Points de terminaison et quotas AWS App Runner dans le document Références générales AWS.

Utilisation des rôles pour la mise en réseau

AWS App Runner utilise des Gestion des identités et des accès AWS rôles liés à un service (IAM). Un rôle lié à un service est un type unique de rôle IAM directement lié à App Runner. Les rôles liés à un service sont prédéfinis par App Runner et incluent toutes les autorisations dont le service a besoin pour appeler d'autres AWS services en votre nom.

Un rôle lié à un service facilite la configuration d'App Runner, car vous n'avez pas à ajouter manuellement les autorisations nécessaires. App Runner définit les autorisations associées à ses

rôles liés aux services et, sauf indication contraire, seul App Runner peut assumer ses rôles. Les autorisations définies comprennent la politique de confiance 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. Cela protège vos ressources App Runner, car vous ne pouvez pas supprimer par inadvertance l'autorisation d'accès aux ressources.

Pour plus d'informations sur les autres services qui prennent en charge les rôles liés à un service, consultez Services AWS qui fonctionnent avec IAM et recherchez les services avec un Oui dans la colonne Rôle lié à un service. Choisissez un Oui ayant un lien permettant de consulter les détails du rôle pour ce service.

Autorisations de rôle liées au service pour App Runner

App Runner utilise le rôle lié au service nommé. AWSServiceRoleForAppRunnerNetworking

Le rôle permet à App Runner d'effectuer les tâches suivantes :

- Connectez un VPC à votre service App Runner et gérez les interfaces réseau.

Le rôle AWSService RoleForAppRunnerNetworking lié à un service fait confiance aux services suivants pour assumer le rôle :

- `networking.apprunner.amazonaws.com`

La politique d'autorisations de rôle nommée `AppRunnerNetworkingServiceRolePolicy`contient toutes les autorisations dont App Runner a besoin pour effectuer des actions en votre nom.

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 en savoir plus, consultez Service-Linked Role Permissions (autorisations du rôle lié à un service) dans le Guide de l'utilisateur IAM.

Création d'un rôle lié à un service pour App Runner

Vous n'avez pas besoin de créer manuellement un rôle lié à un service. Lorsque vous créez un connecteur VPC dans le AWS Management Console, le, ou l' AWS API AWS CLI, App Runner crée 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 connecteur VPC, App Runner crée à nouveau le rôle lié au service pour vous.

Modification d'un rôle lié à un service pour App Runner

App Runner ne vous permet pas de modifier le rôle AWSService RoleForAppRunnerNetworking lié au service. Après avoir 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.

Supprimer un rôle lié à un service pour App Runner

Si vous n'avez plus besoin d'utiliser une fonctionnalité 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. Cependant, vous devez nettoyer votre rôle lié à un service avant de pouvoir le supprimer manuellement.

Nettoyage d'un rôle lié à un service

Avant de pouvoir utiliser IAM pour supprimer un rôle lié à un service, vous devez supprimer toutes les ressources utilisées par le rôle.

Dans App Runner, cela signifie dissocier les connecteurs VPC de tous les services App Runner de votre compte et supprimer les connecteurs VPC. Pour de plus amples informations, veuillez consulter the section called "Trafic sortant".

> **Note**
>
> Si le service App Runner utilise le rôle lorsque vous essayez de supprimer les ressources, la suppression risque d'échouer. Si cela se produit, patientez quelques minutes et réessayez.

Suppression manuelle du rôle lié à un service

Utilisez la console IAM, le AWS CLI, ou l' AWS API pour supprimer le rôle lié au AWSService RoleForAppRunnerNetworking service. Pour en savoir plus, consultez Suppression d'un rôle lié à un service dans le Guide de l'utilisateur IAM.

Régions prises en charge pour les rôles liés au service App Runner

App Runner prend en charge l'utilisation de rôles liés à un service dans toutes les régions où le service est disponible. Pour plus d'informations, consultez Points de terminaison et quotas AWS App Runner dans le document Références générales AWS.

AWS politiques gérées pour AWS App Runner

Une politique AWS gérée est une politique autonome créée et administrée par AWS. AWS les politiques gérées sont conçues pour fournir des autorisations pour de nombreux cas d'utilisation courants afin que vous puissiez commencer à attribuer des autorisations aux utilisateurs, aux groupes et aux rôles.

N'oubliez pas que les politiques AWS gérées peuvent ne pas accorder d'autorisations de moindre privilège pour vos cas d'utilisation spécifiques, car elles sont accessibles à tous les AWS clients. Nous vous recommandons de réduire encore les autorisations en définissant des politiques gérées par le client qui sont propres à vos cas d'utilisation.

Vous ne pouvez pas modifier les autorisations définies dans les politiques AWS gérées. Si les autorisations définies dans une politique AWS gérée sont AWS mises à jour, la mise à jour affecte toutes les identités principales (utilisateurs, groupes et rôles) auxquelles la politique est attachée. AWS est le plus susceptible de mettre à jour une politique AWS gérée lorsqu'une nouvelle politique Service AWS est lancée ou lorsque de nouvelles opérations d'API sont disponibles pour les services existants.

Pour plus d'informations, consultez Politiques gérées par AWS dans le Guide de l'utilisateur IAM.

Mises à jour des politiques AWS gérées par App Runner

Consultez les informations relatives aux mises à jour des politiques AWS gérées pour App Runner depuis que ce service a commencé à suivre ces modifications. Pour recevoir des alertes

automatiques concernant les modifications apportées à cette page, abonnez-vous au flux RSS sur la page d'historique du document App Runner.

Modifier	Description	Date
AWSAppRunnerReadOnlyAccess : nouvelle politique	App Runner a ajouté une nouvelle politique permettant aux utilisateurs de répertorier et d'afficher les informations relatives aux ressources App Runner.	24 févr. 2022
AWSAppRunnerFullAccess : mise à jour d'une politique existante	App Runner a mis à jour la liste des ressources pour l'`iam:CreateServiceLinkedRole` action afin de permettre la création d'un rôle `AWSServiceRoleForAppRunnerNetworking` lié à un service.	8 février 2022
AppRunnerNetworkingServiceRolePolicy : nouvelle politique	App Runner a ajouté une nouvelle politique permettant à App Runner de passer des appels vers Amazon Virtual Private Cloud pour associer un VPC à votre service App Runner et gérer les interfaces réseau pour le compte des services App Runner. La politique est utilisée dans le rôle `AWSServiceRoleForAppRunnerNetworking` lié au service.	8 février 2022
AWSAppRunnerFullAccess : nouvelle politique	App Runner a ajouté une nouvelle politique permettant aux utilisateurs d'effectuer toutes les actions d'App Runner.	10 janvier 2022
AppRunnerServiceRolePolicy : nouvelle politique	App Runner a ajouté une nouvelle politique permettant à App Runner de passer des appels vers Amazon CloudWatch Logs et Amazon CloudWatc	1 mars 2021

Modifier	Description	Date
	h Events pour le compte des services App Runner. La politique est utilisée dans le rôle `AWSServiceRoleForAppRunner` lié au service.	
AWSAppRunnerServicePolicyForECRAccess : nouvelle politique	App Runner a ajouté une nouvelle politique permettant à App Runner d'accéder aux images Amazon Elastic Container Registry (Amazon ECR) depuis votre compte.	1 mars 2021
App Runner a commencé à suivre les modifications	App Runner a commencé à suivre les modifications apportées AWS à ses politiques gérées.	1 mars 2021

Résolution des problèmes d'identité et d'accès à App Runner

Utilisez les informations suivantes pour vous aider à diagnostiquer et à résoudre les problèmes courants que vous pouvez rencontrer lorsque vous travaillez avec AWS App Runner IAM.

Pour d'autres sujets relatifs à la sécurité d'App Runner, consultezSécurité.

Rubriques

- Je ne suis pas autorisé à effectuer une action dans App Runner
- Je souhaite autoriser des personnes extérieures à moi Compte AWS à accéder à mes ressources App Runner

Je ne suis pas autorisé à effectuer une action dans App Runner

S'il vous AWS Management Console indique que vous n'êtes pas autorisé à effectuer une action, contactez votre administrateur pour obtenir de l'aide. Votre administrateur est la personne qui vous a fourni vos informations de AWS connexion.

L'exemple d'erreur suivant se produit lorsqu'un utilisateur IAM nommé `marymajor` essaie d'utiliser la console pour consulter les détails d'un service App Runner mais ne dispose pas des `apprunner:DescribeService` autorisations nécessaires.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 apprunner:DescribeService on resource: my-example-service
```

Dans ce cas, Mary demande à son administrateur de mettre à jour ses politiques pour lui permettre d'accéder à la *my-example-service* ressource à l'aide de l'`apprunner:DescribeService`action.

Je souhaite autoriser des personnes extérieures à moi Compte AWS à accéder à mes ressources App Runner

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 (ACLs), vous pouvez utiliser ces politiques pour autoriser les utilisateurs à accéder à vos ressources.

Pour plus d'informations, consultez les éléments suivants :

- Pour savoir si App Runner prend en charge ces fonctionnalités, consultezComment App Runner fonctionne avec IAM.
- Pour savoir comment fournir l'accès à vos ressources sur celles Comptes AWS que vous possédez, consultez la section Fournir l'accès à un utilisateur IAM dans un autre utilisateur Compte AWS que vous possédez dans le Guide de l'utilisateur IAM.
- Pour savoir comment fournir l'accès à vos ressources à des tiers Comptes AWS, consultez la section Fournir un accès à des ressources Comptes AWS détenues par 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 en savoir plus sur la différence entre l'utilisation des rôles et des politiques basées sur les ressources pour l'accès intercompte, consultez Accès intercompte aux ressources dans IAM dans le Guide de l'utilisateur IAM.

Enregistrement et surveillance dans App Runner

La surveillance joue un rôle important dans le maintien de la fiabilité, de la disponibilité et des performances de votre AWS App Runner service. La collecte de données de surveillance à partir

de toutes les parties de votre AWS solution vous permet de corriger plus facilement une panne, le cas échéant. App Runner intègre plusieurs AWS outils pour surveiller vos services App Runner et répondre aux incidents potentiels.

CloudWatch Alarmes Amazon

Avec les CloudWatch alarmes Amazon, vous pouvez consulter un indicateur de service sur une période que vous spécifiez. Si la métrique dépasse un seuil donné pendant un certain nombre de périodes, vous recevez une notification.

App Runner collecte diverses mesures concernant le service dans son ensemble et les instances (unités de dimensionnement) qui exécutent votre service Web. Pour de plus amples informations, veuillez consulter Métriques (CloudWatch).

Journaux d'application

App Runner collecte le résultat du code de votre application et le diffuse vers Amazon CloudWatch Logs. C'est à vous de décider du contenu de cette sortie. Par exemple, vous pouvez inclure des enregistrements détaillés des demandes adressées à votre service Web. Ces enregistrements de journal peuvent s'avérer utiles dans le cadre d'audits de sécurité et d'accès. Pour de plus amples informations, veuillez consulter Journaux (CloudWatch journaux).

AWS CloudTrail journaux d'actions

App Runner est intégré à AWS CloudTrail un service qui fournit un enregistrement des actions effectuées par un utilisateur, un rôle ou un AWS service dans App Runner. CloudTrail capture tous les appels d'API pour App Runner sous forme d'événements. Vous pouvez consulter les événements les plus récents dans la CloudTrail console et créer un suivi pour permettre la diffusion continue des CloudTrail événements vers un bucket Amazon Simple Storage Service (Amazon S3). Pour de plus amples informations, veuillez consulter Actions d'API (CloudTrail).

Validation de conformité pour App Runner

Des auditeurs tiers évaluent la sécurité et AWS App Runner la conformité de plusieurs programmes de AWS conformité. Il s’agit notamment des certifications SOC, PCI, FedRAMP, HIPAA et d’autres.

Pour savoir si un programme Services AWS de conformité Service AWS s'inscrit dans le champ d'application de programmes de conformité spécifiques, consultez Services AWS la section de conformité et sélectionnez le programme de conformité qui vous intéresse. Pour des informations générales, voir Programmes de AWS conformité Programmes AWS de .

Vous pouvez télécharger des rapports d'audit tiers à l'aide de AWS Artifact. Pour plus d'informations, voir Téléchargement de rapports dans AWS Artifact .

Votre responsabilité en matière de conformité lors de l'utilisation Services AWS est déterminée par la sensibilité de vos données, les objectifs de conformité de votre entreprise et les lois et réglementations applicables. Pour plus d'informations sur votre responsabilité en matière de conformité lors de l'utilisation Services AWS, consultez AWS la documentation de sécurité.

Pour d'autres sujets relatifs à la sécurité d'App Runner, consultezSécurité.

Résilience dans App Runner

L'infrastructure AWS mondiale est construite autour Régions AWS de zones de disponibilité. Régions AWS fournissent plusieurs zones de disponibilité physiquement séparées et isolées, connectées par un réseau à faible latence, à haut débit et hautement redondant. Avec les zones de disponibilité, vous pouvez concevoir et exploiter des applications et des bases de données qui basculent automatiquement d'une zone de disponibilité 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 zones de disponibilité Régions AWS et les zones de disponibilité, consultez la section Infrastructure AWS globale.

AWS App Runner gère et automatise l'utilisation de l'infrastructure AWS mondiale en votre nom. Lorsque vous utilisez App Runner, vous bénéficiez des mécanismes de disponibilité et de tolérance aux pannes qu'il AWS propose.

Pour d'autres sujets relatifs à la sécurité d'App Runner, consultezSécurité.

Sécurité de l'infrastructure dans AWS App Runner

En tant que service géré, AWS App Runner il est protégé par les procédures de sécurité du réseau AWS mondial décrites dans le livre blanc Amazon Web Services : présentation des processus de sécurité.

Vous utilisez des appels d'API AWS publiés pour gérer et faire fonctionner App Runner via le réseau. Les clients qui appellent App Runner APIs doivent prendre en charge le protocole TLS (Transport Layer Security) 1.2 ou version ultérieure. Les clients doivent aussi prendre en charge les suites

de chiffrement PFS (Perfect Forward Secrecy) comme Ephemeral Diffie-Hellman (DHE) ou Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La plupart des systèmes modernes tels que Java 7 et les versions ultérieures prennent en charge ces modes. Ces exigences ne s'appliquent pas aux terminaux des applications App Runner.

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 d'autres sujets relatifs à la sécurité d'App Runner, consultezSécurité.

Utilisation d'App Runner avec des points de terminaison VPC

Votre AWS application peut intégrer AWS App Runner des services à d'autres Services AWS services exécutés dans un VPC à partir d'Amazon Virtual Private Cloud (Amazon VPC). Certaines parties de votre application peuvent envoyer des requêtes à App Runner depuis le VPC. Par exemple, vous pouvez l'utiliser AWS CodePipeline pour effectuer un déploiement continu sur votre service App Runner. L'un des moyens d'améliorer la sécurité de votre application consiste à envoyer ces requêtes App Runner (et à d'autres Services AWS) via un point de terminaison VPC.

À l'aide d'un point de terminaison VPC, vous pouvez connecter en privé votre VPC aux services de point de terminaison Services AWS VPC pris en charge et alimentés par. AWS PrivateLink Vous n'avez pas besoin d'une passerelle Internet, d'un appareil NAT, d'une connexion VPN ou d' Direct Connect une connexion.

Les ressources de votre VPC n'utilisent pas d'adresses IP publiques pour interagir avec les ressources App Runner. Le trafic entre votre VPC et App Runner ne quitte pas le réseau Amazon. Pour plus d'informations sur les points de terminaison VPC, consultez la section Points de terminaison VPC dans le Guide.AWS PrivateLink

> **Note**
>
> Par défaut, l'application Web de votre service App Runner s'exécute dans un VPC fourni et configuré par App Runner. Ce VPC est public. Cela signifie qu'il est connecté à Internet. Vous pouvez éventuellement associer votre application à un VPC personnalisé. Pour de plus amples informations, veuillez consulter the section called “Trafic sortant”.
> Vous pouvez configurer vos services pour accéder à Internet AWS APIs, y compris lorsque votre service est connecté à un VPC. Pour obtenir des instructions sur la façon

d'activer l'accès public à Internet pour le trafic sortant VPC, consultez. the section called "Considérations relatives à la sélection d'un sous-réseau"
App Runner ne prend pas en charge la création d'un point de terminaison VPC pour votre application.

Configuration d'un point de terminaison VPC pour App Runner

Pour créer le point de terminaison VPC d'interface pour le service App Runner dans votre VPC, suivez la procédure de création d'un point de terminaison d'interface décrite dans le guide.AWS PrivateLink Pour Service Name (Nom du service), choisissez `com.amazonaws.region.apprunner`.

Considérations relatives à la confidentialité des réseaux VPC

Important

L'utilisation d'un point de terminaison VPC pour App Runner ne garantit pas que tout le trafic provenant de votre VPC reste hors d'Internet. Le VPC est peut-être public. En outre, il est possible que certaines parties de votre solution n'utilisent pas les points de terminaison VPC pour effectuer AWS des appels d'API. Par exemple, Services AWS vous pouvez appeler d'autres services en utilisant leurs points de terminaison publics. Si la confidentialité du trafic est requise pour la solution de votre VPC, lisez cette section.

Pour garantir la confidentialité du trafic réseau dans votre VPC, tenez compte des points suivants :

- Activer le nom DNS : certaines parties de votre application peuvent toujours envoyer des requêtes à App Runner via Internet en utilisant le point de terminaison `apprunner.region.amazonaws.com` public. Si votre VPC est configuré avec un accès à Internet, ces demandes aboutissent sans aucune indication pour vous. Vous pouvez éviter cela en vous assurant que l'option Activer le nom DNS est activée lorsque vous créez le point de terminaison. Par défaut, il est défini sur true. Cela ajoute une entrée DNS dans votre VPC qui mappe le point de terminaison du service public au point de terminaison de VPC d'interface.
- Configurer les points de terminaison VPC pour des services supplémentaires : votre solution peut envoyer des demandes à d'autres personnes. Services AWS Par exemple, AWS CodePipeline peut envoyer des demandes à AWS CodeBuild. Configurez les points de terminaison VPC pour ces services et activez les noms DNS sur ces points de terminaison.

- Configuration d'un VPC privé : si possible (si votre solution n'a pas du tout besoin d'accès à Internet), configurez votre VPC comme privé, ce qui signifie qu'il ne dispose pas de connexion Internet. Cela garantit qu'un point de terminaison VPC manquant provoque une erreur visible, afin que vous puissiez ajouter le point de terminaison manquant.

Utilisation des stratégies de point de terminaison pour contrôler l'accès avec des points de terminaison de VPC

Les politiques de point de terminaison VPC sont prises en charge pour App Runner. Par défaut, l'accès complet à App Runner est autorisé via le point de terminaison de l'interface. Les politiques de point de terminaison VPC peuvent être utilisées pour contrôler les principaux AWS autorisés à accéder au point de terminaison App Runner. Vous pouvez également associer un groupe de sécurité aux interfaces réseau du point de terminaison pour contrôler le trafic vers App Runner via le point de terminaison de l'interface.

Intégration au point de terminaison de l'interface

App Runner prend en charge AWS PrivateLink, ce qui fournit une connectivité privée à App Runner et élimine l'exposition du trafic à Internet. Pour permettre à votre application d'envoyer des demandes à App Runner à l'aide de AWS PrivateLink, configurez un type de point de terminaison VPC appelé point de terminaison d'interface. Pour plus d'informations, consultez la section Interface VPC endpoints (AWS PrivateLink) dans le Guide.AWS PrivateLink

Analyse de configuration et de vulnérabilité dans App Runner

AWS et nos clients partagent la responsabilité d'atteindre un niveau élevé de sécurité et de conformité des composants logiciels. Pour plus d'informations, consultez le modèle de responsabilité AWS partagée.

Images du conteneur de correctifs

L'application de correctifs à l'image du conteneur fait partie de la responsabilité du client dans le modèle de sécurité partagé. Le propriétaire de l'image est chargé de mettre à jour et de corriger régulièrement l'image du conteneur. Nous vous recommandons d'établir un calendrier de routine pour vérifier et appliquer des mises à jour aux images de vos conteneurs. Pour plus d'informations sur la façon de scanner vos images pour détecter les vulnérabilités, consultez la documentation d'AWS App Runner

Pour d'autres sujets relatifs à la sécurité d'App Runner, consultezSécurité.

Bonnes pratiques de sécurité pour App Runner

AWS App Runner fournit plusieurs fonctionnalités de sécurité à prendre en compte lors de l'élaboration et de la mise en œuvre de vos propres politiques 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 considérations utiles et non comme des recommandations.

Pour d'autres sujets relatifs à la sécurité d'App Runner, consultezSécurité.

Bonnes pratiques de sécurité préventive

Les contrôles de sécurité préventifs tentent d'éviter les incidents avant qu'ils ne se produisent.

Implémentation d’un accès sur la base du moindre privilège

App Runner fournit des politiques gérées Gestion des identités et des accès AWS (IAM) pour les utilisateurs IAM et le rôle d'accès. Ces politiques gérées spécifient toutes les autorisations qui peuvent être nécessaires au bon fonctionnement de votre service App Runner.

Votre application peut ne pas exiger toutes les autorisations de nos stratégies gérées. Vous pouvez les personnaliser et n'accorder que les autorisations nécessaires à vos utilisateurs et à votre service App Runner pour effectuer leurs tâches. C'est particulièrement pertinent pour les stratégies utilisateur, où différents rôles utilisateur peuvent avoir des besoins différents en matière d'autorisations. L’implémentation d’un accès sur la base du moindre privilège est fondamentale pour réduire les risques en matière de sécurité et l’impact que pourraient avoir des d’erreurs ou des actes de malveillance.

Analysez vos images pour détecter les vulnérabilités

Vous pouvez utiliser les Amazon ECR APIs pour identifier les vulnérabilités logicielles dans les images de vos conteneurs. Pour plus d'informations, consultez la documentation Amazon ECR.

Bonnes pratiques de sécurité de détection

Les contrôles de sécurité de détection identifient les violations de sécurité après qu'elles se sont produites. Ils peuvent vous aider à détecter une menace ou un incident de sécurité potentiel.

Mise en œuvre de la surveillance

La surveillance joue un rôle important dans le maintien de la fiabilité, de la sécurité, de la disponibilité et des performances de vos solutions App Runner. AWS fournit plusieurs outils et services pour vous aider à surveiller vos AWS services.

Voici quelques exemples d'éléments à surveiller :

- CloudWatch Métriques Amazon pour App Runner : définissez des alarmes pour les indicateurs clés d'App Runner et pour les indicateurs personnalisés de votre application. Pour en savoir plus, consultez Métriques (CloudWatch).
- AWS CloudTrail entrées — Suivez les actions susceptibles d'avoir un impact sur la disponibilité, comme `PauseService` ou`DeleteConnection`. Pour en savoir plus, consultez Actions d'API (CloudTrail).

AWS Glossaire

Pour la AWS terminologie la plus récente, consultez le AWS glossaire dans la Glossaire AWS référence.

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.